AD-A268 102

[DOCU]MENTATION PAGE

Form Approved
OMB No 0704-0188

[Public reporting burden for this collection of information is e]stimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, [gathering] and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this [collection of information, including suggestions for reducing] this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson [Davis Highway ...] the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1993	THESIS/DISSERTATION

4. TITLE AND SUBTITLE
THREE DIMENSIONAL VISUALIZATION OF GOES CLOUD DATA USING OCTRESS

5. FUNDING NUMBERS

6. AUTHOR(S)
Capt Bruce H. VanAartsen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
AFIT Student Attending: Colorado State University

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/CI/CIA- 93-119

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DEPARTMENT OF THE AIR FORCE
AFIT/CI
2950 P STREET
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release IAW 190-1
Distribution Unlimited
MICHAEL M. BRICKER, SMSgt, USAF
Chief Administration

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

93 8 16 004

93-19025

14. SUBJECT TERMS	15. NUMBER OF PAGES
	79
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

THESIS

THREE-DIMENSIONAL VISUALIZATION OF GOES CLOUD DATA USING OCTREES

Submitted by

Bruce H. Van Aartsen

Department of Atmospheric Science

In partial fulfillment of the requirements

for the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 1993

COLORADO STATE UNIVERSITY

May 10, 1993

WE HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER OUR SUPERVISION BY BRUCE H. VAN AARTSEN ENTITLED THREE-DIMENSIONAL VISUALIZATION OF GOES CLOUD DATA USING OCTREES BE ACCEPTED AS FULFILLING IN PART THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

Committee on Graduate Work

Adviser

Department Head

ACKNOWLEDGEMENTS

First of all I must thank my wife, Lisa, for her patience, love, and support though these last two years of sometimes grueling work toward completion of my Master's degree. I have no doubt that her tasks at home with our three young children were often tougher than my tasks here at school.

I would like to thank my advisor, Dr. Thomas Vonder Haar, for his guidence in this work. His direction and confidence in me provided both technical and moral support needed for completion of this thesis. Thanks are also due to committee members Dr. Thomas McKee and Dr. Thomas Brubaker for corrections and inputs to making the final version easier to read and interpret.

Special thanks go to Nan McClurg for all her computer system help, and for quick response to my numerous requests at inconvenient hours. Special thanks also to Cindy Combs for many hours of tutoring on GOES data processing and reduction, and to Jan Behunek for sharing his expertise on the GOES/IMX data format and his GOES conversion routine. Of course, much thanks must also be given to Russell Huonder for writing the VOXEL program that this thesis used to examine the octree structure.

The following people have helped with discussion and technical assistance during the past two years: Andy Jones, John Forsythe, and Harold Gibson (office mates), Kelly Dean, Mike Hiatt, Duane Whitcomb, Don Reinke, Dave Randel, and Ken Eis. Many thanks to all for your time and contributions to this effort. Also thanks are due to Loretta Wilson for her willing administrative support throughout this project.

Finally, I must thank Maj Mary Smith, my AFIT program manager. She was instrumental in getting me off "The Rock" (Shemya AFB, AK) without a threatened two-month extension, so I could rejoin my family here at CSU. For this I will be eternally grateful. I thank Dr. McKee a second time for his role in this reprieve.

ABSTRACT OF THESIS

THREE-DIMENSIONAL VISUALIZATION OF GOES CLOUD DATA USING OCTREES

Geostationary Operational Environmental Satellite (GOES) imagery is commonly displayed in a two-dimensional format. This is adequate to portray areas as being simply cloudy or clear, but often more information is desired. A three-dimensional display could be helpful in analyzing the information, and could improve understanding of the atmospheric conditions. Many three-dimensional visualization packages model just the smoothed object surfaces. While this may be aesthetically pleasing, it does not effectively represent volumetric data, and the smoothed surfaces may compromise data integrity.

This thesis presents *octree encoding* as a structure that can effectively represent volumetric data with the added benefits of data compression and acceleration of volume rendering. The fundamentals of octree encoding are introduced, along with strengths and weaknesses, and which data sets are better suited to octree compression.

To demonstrate the capabilities of octree encoding of cloud data, GOES imagery was gathered over Central America for the month of December 1990, and a program called VOXEL[1] (VOlumetric piXEL) was utilized for octree encoding and three-dimensional visualization. VOXEL displays are shown to illustrate the greater visual impact of three-dimensional vs. two-dimensional data. VOXEL is also used to demonstrate how two-dimensional cloud climatologies are overlaid to create a three-dimensional array, which can then be rotated, sliced, or thresholded as desired. This three-dimensional octree-encoded

[1] VOXEL was written by R.J. Huonder for his Master's Thesis (1990).

data can be a valuable asset to meteorological analysis, and can render the data in less time with smaller data storage requirements.

Bruce H. Van Aartsen
Department of Atmospheric Science
Colorado State University
Fort Collins, Colorado 80523
Summer 1993

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

Figure or Table Number and Title | Page No.

Chapter 1

INTRODUCTION

The three-dimensional octree structure was developed in 1978 by Hunter, who merely expanded on the idea of the two-dimensional quadtree structure. The quadtree structure allows efficient storage of two-dimensional images by recursively dividing square arrays of data into four more squares, and then checking each square for homogeneity. If the data in a square is homogeneous, the entire square is stored as one byte of data. Otherwise quadtree continues subdividing the square until all the data in each smaller square is found to be homogeneous, which may not occur until the resolution of the image is reached. Figure 1.1 shows an example of this technique. Since each homogeneous square is stored as one byte, data storage efficiency is enhanced when large squares of uniform data are found.

The octree structure simply extends the quadtree idea to three dimensions, subdividing cubes of volumetric data into eight smaller cubes, and storing homogeneous cubes of data as a single byte. In this manner, an octree can represent any three-dimensional volume as a collection of cubes while still retaining the interior structure of the volume. An example of octree encoding is shown in Chapter 2.

Due to the efficiency of the octree data structure, its use is becoming more widespread. Octree representation seems to be most heavily used in CAD/CAM systems where some systems generate octrees from model object silhouettes (Ahuja and Veenstra, 1989; Arnand and Knott, 1991) and others, directly from digital stereo images (Li, 1992). Conti (*et. al.*, 1991) has even developed a mutant octree structure for CAD of integrated circuits that can subdivide the cubes into more complex polyhedrons.

Medical imaging is also taking advantage of the octree structure to encode their three-dimensional data (Meagher, 1984; Hull, *et. al.*, 1990). Octrees are well suited to this

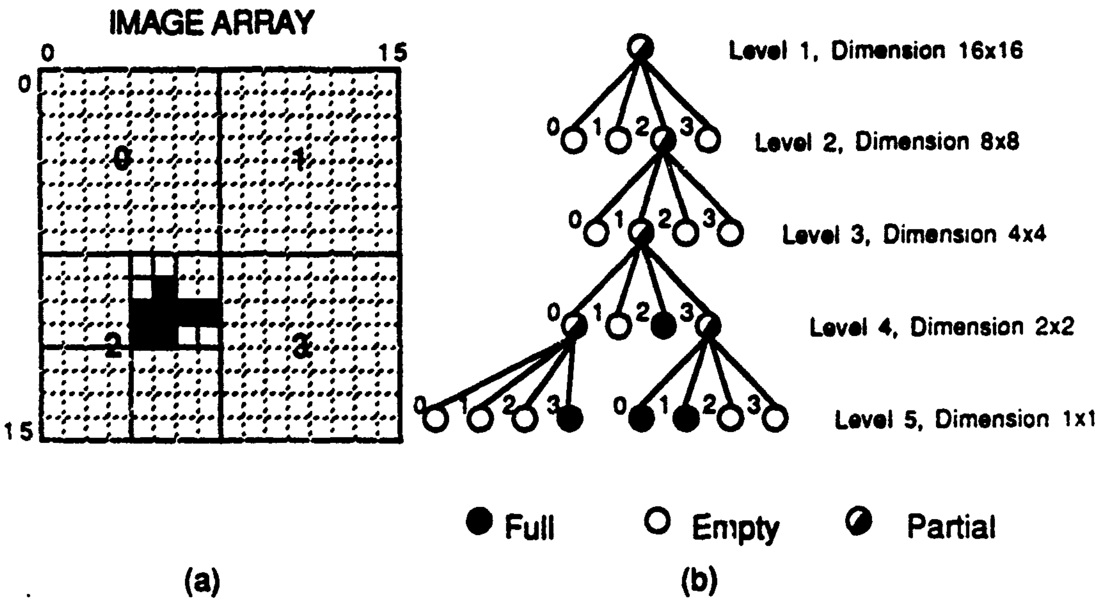

Figure 1.1: Example of a quadtree encoding. (a) The image to be encoded, and (b) the tree structure representing the data. The numbers in the large squares represent the order in which all subsquares are processed. [After Huonder (1990).]

application, since the internal structures can be easily viewed by making an arbitrary slice into the volume of interest.

Octrees have even found an application in robotics (Faverjon, 1984; Ahuja and Veenstra, 1989). The efficient octree structure is used to model the environment robotic arms work in, and allows quick calculation of collision-free paths for the arms.

Recently, Huonder (1990) explored the use of octrees in displaying three-dimensional weather radar and infrared satellite cloud data by using an original software package (VOXEL). This thesis makes use of his VOXEL program to further investigate the usefulness of the octree structure in analyzing GOES cloud data. In Chapter 2, the basics of octree encoding are presented, along with its strengths and weakness, and which data sets are better suited to the octree structure. Chapter 3 introduces the powerful analysis features of VOXEL that were used to display all the 3-D data sets in this thesis. GOES cloud data are then presented in several formats in Chapter 4. In particular, December

cloud data are then presented in several formats in Chapter 4. In particular, December 1990 cloud climatologies are shown, along with other cloud data that demonstrate the benefits of 3-D octree visualization, and how it can help improve our understanding of atmospheric phenomena.

Chapter 2

THE OCTREE STRUCTURE

2.1 Octree Encoding Basics

Octree representations describe objects by a spatial numerical structure. Octree subcubes approach geometric details of objects hierarchically. This hierarchical tree is made up of nodes containing a data value, and eight pointers, hence the name *octree*. The data value specifies the type of data inside the cube. If the cube is filled with homogeneous data, the pointers are set to *nil*. Otherwise, the pointers reference eight children nodes (subcubes), which subsequently are checked for homogeneity.

A universal cube is the original block which contains the object to be described by the octree representation. As shown in Figure 2.1, the universal cube is divided into eight subcubes by halving it in three directions. Each subcube is then checked to see whether it is occupied by the object. The subcubes are classified into three categories: (a) Full (completely occupied by the object); (b) Empty (no object elements in the subcube); or (c) Partial (the subcube is partially occupied by the object). Each subcube classified as Partial is further subdivided until all cubes are either Full or Empty.

Figure 2.2 shows the octree obtained from encoding Figure 2.1. This simple example only requires three levels to encode since the dimension of the smallest cube was only one quarter the dimension of the universal cube. In level one the universal cube is found to be partially occupied, and is divided in to eight subcubes. Of these eight cubes, 0, 1, and 5 are classified Empty, and 2, 6, and 7 are classified Full. Cubes 3 and 4 are found to be Partial, so are further subdivided in the third level. Using the same numbering scheme, cube 3 classifies subcubes 0, 1, 2, 4, and 5 as Empty, and subcubes 3, 6, and 7 as Full. Similarly,

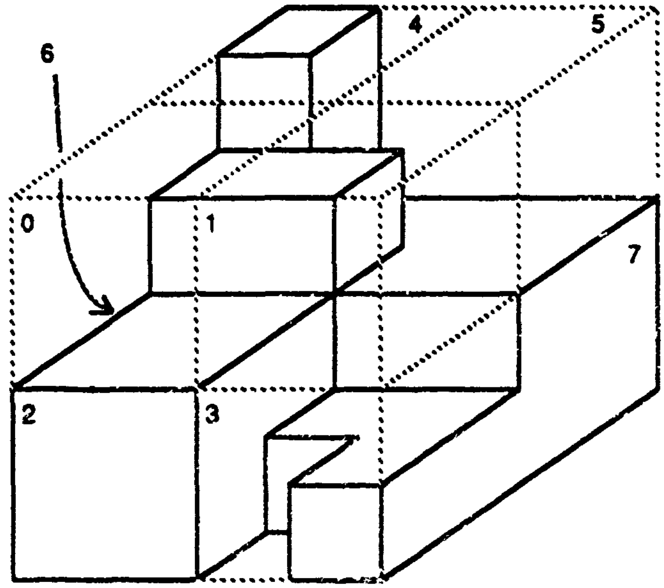

Figure 2.1: Original object to be encoded with the universal cube surrounding object. [After Huonder, 1990]

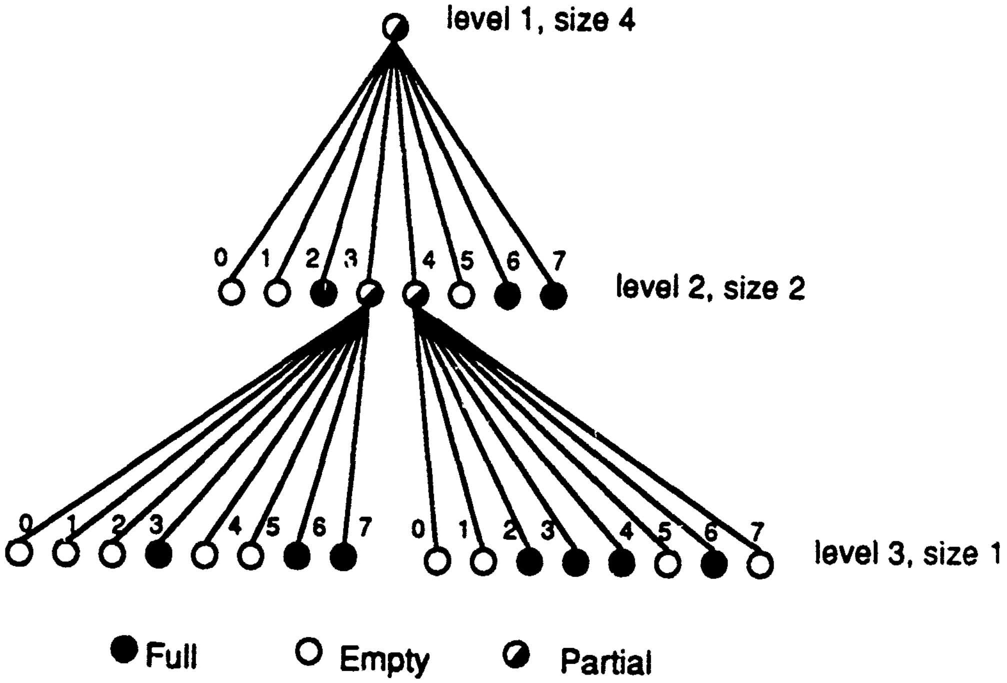

Figure 2.2: Octree structure of encoded data [After Huonder, 1990]

cube 4 classifies subcubes 0, 1, 5, and 7 as Empty, and subcubes 2, 3, 4, and 6 as Full. Since all subcubes were found to contain homogeneous data, the encoding process terminates at this level.

2.2 Advantages and Disadvantages of Using Octrees

The octree structure obviously has several good characteristics. One of the first that should be mentioned is its simplicity. With enough subdivisions, the simple cube can be used to build a representation of the most complex objects. And since just this one primitive building block is used, only one simple set of recursive algorithms is needed to process any data set.

Another previously mentioned advantage is that an object's *volume* is represented, rather than just the object's *surface*. Many visualization packages use polygons to render the surface of an object, but do not retain the inner volume of data. One recent example from a Denver weather broadcast depicted a three-dimensional storm system moving over Colorado's Front Range. But when the view was rotated to see below the bottom edge of the clouds, the inside of the system appeared as a large hollow dome! Though it may have been possible to add cloud bases to the visualization, this was apparently too much effort using their geometric model. Octrees can be utilized to easily visualize the entire volume and inner structure of an object.

The octree structure also has a data storage advantage over the common three-dimensional Cartesian array. This is apparent in that octree can store large cubes of homogeneous data as a single byte, whereas the Cartesian system requires one byte for every element in the array. Though this savings varies with the data set used, one set used in this thesis (see Fig. 4.35) was reduced from 513 blocks of memory in Cartesian form, to 74 blocks of memory in octree form. One side benefit to the larger cubes is that they also take less time to draw on the screen.

While the octree structure has an advantage over Cartesian arrays, it can still require huge amounts of memory for high resolution applications. The memory needed increases

exponentially with the levels of subdivision, and is proportional to the surface area of the represented object (Huondar, 1990). State-of-the-art medical applications may go up to 20 levels of subdivisions, using millions of octree nodes, to create nearly smooth renderings of entire human heads with the complete inner structure (Wilhelms and Van Gelder, 1992).

2.3 Data Sets Best Suited to Octree Encoding

Although high resolution applications may require vast amounts of memory, GOES cloud data is well suited to octree encoding since visible channels can do no better than 1x1 km[1] resolution, and infrared is 3x7 km[2] at best. As such, only expansive data sets would need octrees of more than 10 or 12 levels. This thesis uses 8-level octrees which is adequate for the infrared data sets used.

Since octrees are most efficient when large blocks of homogeneous data are present, its best use is on smooth-surfaced objects with a highly-ordered inner structure. If data sets that are more chaotic are encoded, fewer large homogeneous blocks will be found, and octree encoding will be less efficient. This principle was demonstrated in this thesis when data from both climatologies and single occurrences were encoded. The clouds from a single infrared image were naturally well-ordered, while the climatological data sets were slightly chaotic. Depending on the type of display, the octree encoded climatology required from 2 to 6 times as much memory as the octree of the single infrared image.

---

[1] Resolution is 0.75 x 0.86 km at nadir

[2] Resolution is 3.00 x 6 87 km with oversampling at nadir.

Chapter 3

CAPABILITIES OF THE VOXEL SOFTWARE PACKAGE

The VOXEL (VOLumetric piXEL) program was written by Russell J. Huonder (1990) to make it simple for anyone to take advantage of the power of octree encoding and visualization. The user-friendly program runs on X-windows using a menu-driven graphic interface controlled by a three-button mouse. The data input must be a three-dimensional array[a], which VOXEL will encode to octree format, and then save to file for future use. Following is a description of all VOXEL's major visualization features.

3.1 ROTATION AND ZOOMING

In order to examine areas of interest in the data set, both zooming (magnification) and arbitrary rotation can be set. The rotation menu displays the orientation of the universal cube with sides labeled E, W, N, S, T, and B for east, west, north, south, top, and bottom, respectively. The cube can be rotated along all three of its major axes, as well as along the X, Y, and Z axis of the screen. Rotation increments of 10 degrees, 5 degrees, or 1 degree are controlled by the three mouse buttons. The light source may also be rotated to control shading of the object.

The zoom factor can enlarge or shrink the displayed octree. The allowed range is 1 to 20 with a default value of 5, though trial and error must be used to find the appropriate setting. After a new zoom factor is set, the entire octree must be redrawn to see if the desired magnification has been achieved.

---

[a]Maximum overall dimension is limited to X*Y*Z <= 262144 (i.e. 64*64*64).

3.2 CLIPPING AND THRESHOLDING

Two extremely powerful viewing tools built into the VOXEL program are the clipping and thresholding options. These tools allow viewing the internal structure of an octree. Each of the nodes in the octree holds a data value of 0 to 255 and the thresholding option lets nodes that are above or below a specified value be excluded from the display. In medical applications this allows only mass above a certain density to be displayed, while in this thesis, it could strip away clouds that appeared below a set frequency-of-occurrence.

Just as powerful is the clipping tool. This option permits the octree to be sliced along any of the three axes, clipping away the exterior and revealing the object's cross-section at any arbitrary height, width, or depth. This was especially helpful in this thesis for viewing vertical distributions of cloud frequencies.

Both of these tools are easy to use, and each has their own strengths in analyzing the areas of interest inside an octree volume.

3.3 IMAGE LOOPING

Since it can take several seconds (or even minutes[4]) to redraw large complex octrees, having several frames to draw to can be a great convenience. Though VOXEL can only have one octree in memory at a time, it can retain up to eight octree images. After drawing one octree view on the first frame, one can elect to have the next view drawn to a different frame. After all desired views have been drawn to the frames, the looping feature can be used to flip through any or all of the eight frames available. Frames can be chosen in any order and looped through both forwards and backwards.

If any of the frames is wanted for future reference, it can be saved to a file and reloaded later. Printing hard copies cannot not be done directly from the VOXEL program, but the

---

[4] Depending on the speed of the workstation used.

files are saved to a format that can also be loaded into IMX[5] (Image Manager for X-windows). From IMX, appropriate color tables and graphic overlays can be added to the images and the final product can be output to a color printer.

3.4 DATA FUSION

Though VOXEL does not have any special features set aside for merging of data sets, it does provide for an effective display of previously merged data. By setting different data sets to different ranges of color table values in the original data array, VOXEL's color editing options can be used to differentiate between the data sets. VOXEL's eight colors can be set to any range of values between 0 and 255, and arranged in any order desired. In this thesis, that allowed setting GOES imagery from different valid times to different colors, so that changes from one image to the next would be readily apparent.

Another means of overlaying the data sets is by loading different data sets into VOXEL and then saving the octree images of the data using the same viewing angle. These saved images can then be loaded into IMX where a voxel block "wire frame" can be made from one image and then overlaid on the other image. This technique works best at higher zoom factors, since too many small blocks make the visualization appear too "busy".

[5] IMX was developed by CIRA (Cooperative Institute for Research in the Atmosphere) at Colorado State University.

Chapter 4

VISUALIZATION OF OCTREE-ENCODED GOES CLOUD DATA

To demonstrate the usefulness of three-dimensional octree visualization, two types of data sets were assembled. For the first type, major work was involved compositing sets of images to compile cloud climatologies, while the other type of data displayed is from single infrared images. Both types of data were assembled using the same original data set.

4.1 DATA COLLECTION

The data used in this thesis was extracted from CIRA's vast archive of satellite data. The original data set was gathered during the entire month of December 1990, using the Colorado State University (CSU) Interactive Satellite Ingest System (ISIS). The archived set was found to have 25 days of usable GOES data, with 15 and 18 GMT images available in both visible (VIS) and infrared (IR) format[6]. Both formats are utilized in this study.

The data set covers Central America, stretching from Guatemala to Columbia. This sector was chosen due to interest by the DOD in that area, and personal curiosity about the diurnal and orographic effects in that region. As the climatologies will show, the Inter-tropical Convergence Zone (ITCZ) also has a major effect on the chosen sector. In fact, the data set was switched from to December from July early in the research, since the ITCZ caused nearly constant overcast conditions over much of the sector, leaving no landmarks for realignment of the images.

---

[6]Visible wavelength of $6.7*10^{-6}$ m. and infrared wavelength of $11.2*10^{-6}$m.

An additional advantage of this data sector is its proximity to GOES' nadir. This not only means near optimum resolution, but also parallax errors are kept to a minimum.

4.1.1 CONVERTING RAW DATA TO IMX-VIEWABLE FORMAT

The original satellite data for this study included GOES VIS and IR channels at full resolution which had been archived to tape without any format conversion. After first transferring all 100 images from tape to disk, the data had to be put into IMX format using a routine called ISISGOES. This still yielded an image too large to view, however, and a windowing routine was initially used to lower the resolution so that the image would fit inside IMX's maximum display dimensions of 1024 x 1024 pixels (picture elements). The original sector was quite large, covering a roughly square area between southern Florida and northern Peru (25° N. to 5° S.) and the windowed view was used to help choose a specific sector to study.

Since VOXEL could not encode an image larger than 128 x 128 pixels[7], a small enough area had to be chosen so that resolution would not be greatly degraded by converting the image to VOXEL format. The Central American sector chosen was centered at 11.5° N and 83.3° W with a structure defined to be 1024 x 900 pixels at two kilometer resolution. This resulted in a sector that was approximately 18° of longitude by 16° of latitude. The SECTOR routine from CSU's Interactive Research and Imaging System (IRIS) was utilized to extract the desired area and resolution from each of the images.

4.1.2 PUTTING GOES DATA INTO VOXEL-READABLE FORMAT

Converting GOES/IMX images into VOXEL input arrays requires in-depth knowledge of both formats. Using FORTRAN, each line from the image must be directly read into a single-dimensioned BYTE array with as many elements as there are pixels in the line.

[7] With a maximum vertical coordinate of 16.

Then each of the elements in the array can be processed individually to discover the brightness count value. This value could then be dealt with as desired (i.e., converted to temperature or height).

Before outputting to VOXEL array, it must be decided if the value from the image is to be represented as a color value or depicted topographically in the octree. To create a topographical portrayal, a representative "stack" of elements in the 3-D array must be set to a non-zero integer. To use color representation, the value from the image must be set to an appropriate color table intensity, and that integer then input into the VOXEL array. In both cases, the integer value that is put into the array must be between 0 and 253, which is the range of color intensities valid in the VOXEL program. For an effective octree display, both color and topography are used to represent different aspects of the data.

Fortunately, a portion of the conversion source code (named GOESVOX) had already been written at CIRA[*], which was then appended and modified to output various types of octree displays. Though not implemented in this thesis, existing subroutines can compute cloud bases from rawinsonde data, allowing realistic high-resolution cloud representations.

4.2 CLOUD CLIMATOLOGY OVER CENTRAL AMERICA

In order to compile a valid cloud climatology, all the images had to be renavigated to correct for errors in navigation parameters or for satellite wobble. The first step in this process was to begin with a base VIS image from a nearly cloudless day. This image was used to draw a base graphic on IMX, showing all the coastlines and large lakes. The other VIS images were then displayed on IMX and panned around until the coastlines matched the graphic overlay. Subsequently, the IR images were panned the same amount as their VIS counterparts. In this manner, all the images could be near-perfectly aligned. After renavigation, all the images were saved to disk for further processing.

---

[*]Jan Behunek wrote the GOES/IMX image decoding source code.

Cloud climatologies rely on brightness thresholding to decide whether or not a cloud occupies a given position of the image. Since brighter land masses can contaminate this thresholding technique, background images were constructed for each data format and valid time. To produce background images, a subroutine was used that read in all 25 images and saved the darkest count value from each pixel position. The subroutine then output a composite of these "hot" pixels, creating an essentially cloud-free image (see Figure 4.1). The background images were then used in another subroutine to compare with all the other images. If the brightness of a certain pixel was greater than the background by a specified threshold amount, a cloud was registered at that pixel location. As shown in Figure 4.2, areas in the climatology that appear brighter represent areas where clouds occurred more frequently at that valid time.

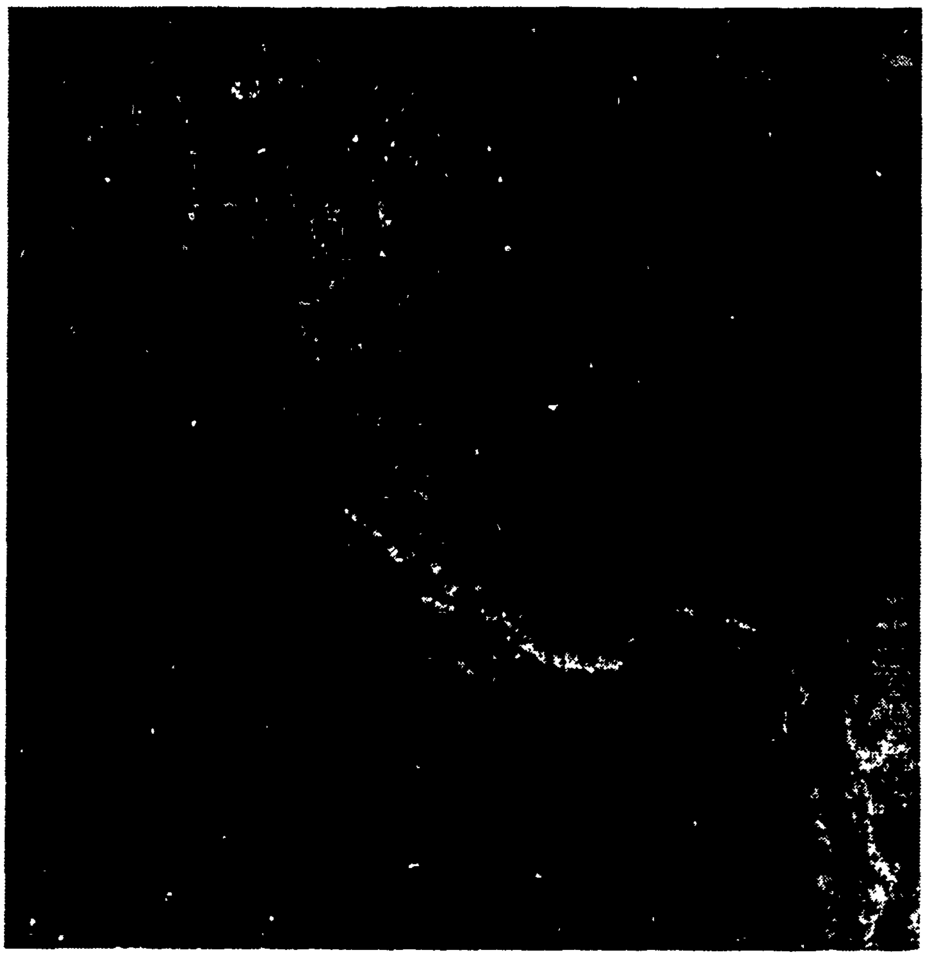

Figure 4.1: Background image from composite of darkest pixels in 18Z December 1990 visible images.

Figure 4.2: Cloud climatology of 18Z December 1990 visible images.

4.2.1 VISUALIZATION OF A SINGLE-THRESHOLD CLIMATOLOGY

An interesting first example of the powerful visual impact of octree visualization is shown in Figure 4.3. This figure is merely a three-dimensional representation of Figure 4.2, with a color table added, making the climatology both easier to comprehend and to interpret. Though a color table could be added to the two-dimensional image, the layman may be able to better relate frequency-of-occurrence to the topographic height coordinate of the 3-D octree display.

In order to construct this 3-D depiction, the IMX image shown in Figure 4.2 had to first be windowed down to 128 x 128 pixels, which are the largest dimensions VOXEL allows. IMX handles this windowing by averaging the values of all the pixels that will be combined into a single pixel. This smaller image was then input to the GOESVOX conversion program, which read the color intensity of each pixel. Since the (gray) color table of the composite was scaled from 0 to 100, GOESVOX was programmed to rescale this range to a maximum of 16 (to stay within VOXEL's maximum overall dimension limits). A GOESVOX routine would then fill the height coordinate of the VOXEL array with the appropriate number of non-zero values to represent the intensity of the pixel. During this process, the non-zero values were set to different color intensities depending on the value of the height coordinate.

To make the visualization appear more three-dimensional, VOXEL's rotation feature was used. The octree in Figure 4.3 was tilted and rotated enough to make three faces of the cubes visible. This view shows the octree as viewed from the south-southwest.

One drawback of using eight color levels is that the non-zero octree cubes never get larger than a dimension of two, since the homogeneous volumes never get any taller than that. Altering GOESVOX to create the octree with four color levels (cubes had a maximum dimension of four) reduced the memory required from 165 blocks to 150 blocks. Though this is a significant savings, both octrees were encoded from 513 block Cartesian arrays, so the biggest savings came from the encoding of large volumes of (homogeneous) zero-valued data from the empty spaces in the octree.

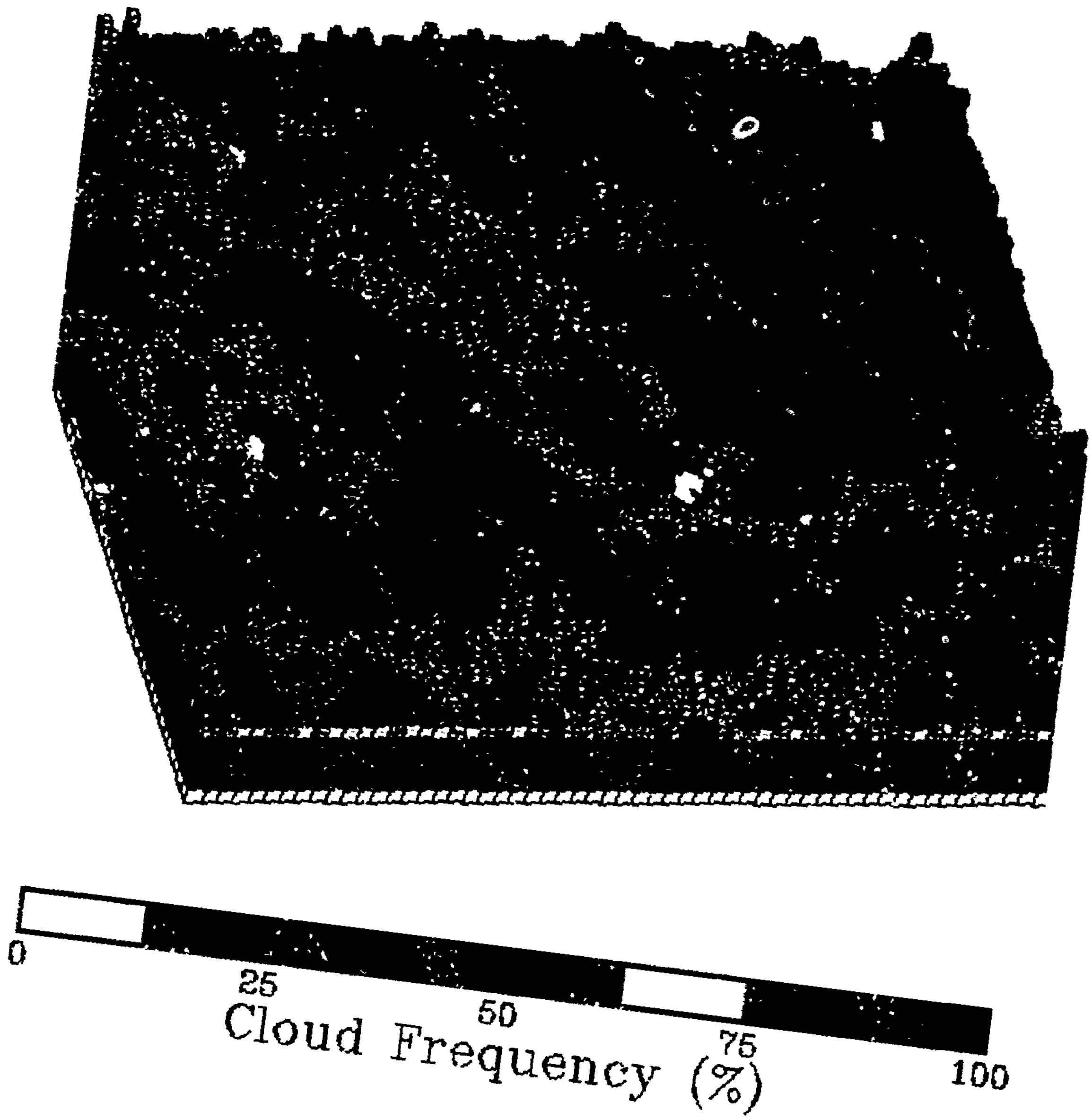

Figure 4.3: Cloud climatology from 18Z December 1990 VIS images in three-dimensional topographic octree format.

It is difficult to discern where coastlines and geopolitical boundaries are from the display in Figure 4.3. One way of remedying this is shown in Figure 4.4. This figure portrays borders and coastlines as clear blocks with white edges, which have been overlaid at the 16th (top) level of the octree. Though this method masks some of the data, it does give a good idea of the boundary positions without referring to Figure 4.2 and then making an educated guess as to where the boundaries lie.

The wire frame overlay effect was accomplished with the aid of IMX. First the IMX graphic overlay was converted to an IMX image with 128 x 128 resolution. This image was then input to GOESVOX which was programmed to output the graphic to the top level of a 128 x 128 x 16 octree. VOXEL was then used to display the octree-encoded graphic, using the same magnification and rotation as the previously-saved 3-D climatology. The VOXEL graphic was then saved to an image file and loaded back into IMX, where just the black box-edges were converted to a graphic overlay. This octree-cube overlay was then added to the 3-D climatology to produce the transparent cubes seen in Figure 4.4 (the background was also changed to blue so that all the white "wire frame" could be seen).

4.2.2 THREE-DIMENSIONAL CLIMATOLOGY USING MULTIPLE THRESHOLDS

The previous section looked at a climatology from a single threshold. If several climatologies from different thresholds are overlaid, a truly three-dimensional data set is obtained. This method was used on both 15Z and 18Z IR climatologies. Using a higher threshold (above background brightness) means only cooler/higher clouds show up on the climatology. Consequently, overlaying higher and higher thresholds creates a 3-D data set with a pseudo-height coordinate.

To create this type of octree, GOESVOX was modified to read in 16 images with increasing threshold values. Each threshold was output to a single level of the octree, using color intensity values to represent frequency of cloud occurrence. The resulting 16-level octree can be clipped or thresholded to analyze different aspects of the data set. Figures 4.5 through 4.18 show the first 14 levels of thresholding of the 18Z climatology and were easily

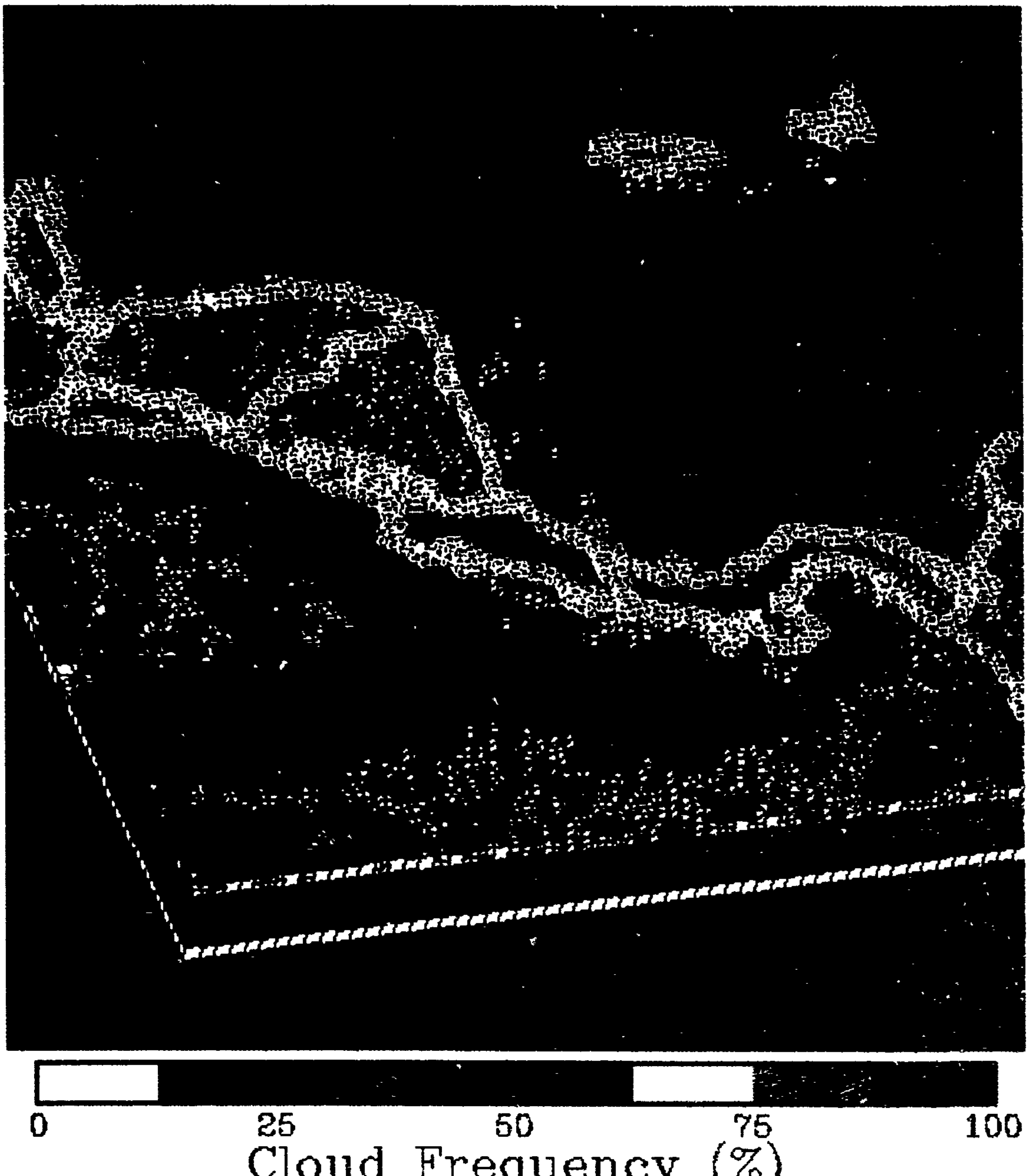

Figure 4.4: 3-D topographic cloud climatology with octree-cube graphic overlay.

displayed by clipping the octree along each of the lower 14 planes. The top two levels are not shown since they both appeared totally white. For comparison purposes, Figures 4.19 through 4.24 show selected thresholds from the 15Z data set. Graphic overlays and color tables were added using IMX.

Since larger volumes of homogeneous data are octree encoded as larger cubes, VOXEL represents this by drawing larger squares in the figures. For example, in Figure 4.24 there is a square over Guatemala that has sides eight times as large as the smallest squares. This means there was no change in cloud frequency-of-occurrence throughout an entire cube of data (8 x 8 x 8). This figure shows the tenth level of sixteen levels, so the entire cube must have occupied level nine through level sixteen. The actual cloud frequency was 0% in this large cube, though this is not evident from the color table which only indicates a value between 0 and 12.5%.

While clipping the octree at the horizontal plane of interest is a useful tool, the power of octree displays is more fully utilized when examining the three-dimensional structure of a data set. To investigate the vertical structure of the 15Z climatology, several vertical cross-sections were taken (Figures 4.25-4.28). Note that in these figures, the white cubes display topographic features due to a threshold setting of 1 (if no clouds were ever present at a certain position, no cube was drawn). Of particular interest is the cross-section in Figure 4.28, which shows a dramatic increase in mid/upper level cloud frequencies to the east of Nicaragua, while at the same time, a *decrease* is apparent in the lowest level for this same region.

For better analysis, the Nicaraguan region was cut out and further magnified in Figures 4.29-4.31. In these figures, VOXEL's thresholding feature was utilized to improve comprehension of the height/frequency relationship. Figure 4.29 shows the area with a cloud frequency threshold set at 1%, while Figures 4.30 and 4.31 exclude cloud frequencies below 12.5% and 25% respectively. From this type of display, it is readily apparent that the taller storms are generally located over the ocean east of Nicaragua, and the clouds that occur over the land mass have little vertical extent.

To make 3-D multi-thresholded climatologies useful to pilots, nominal temperatures and/or heights could be assigned to each threshold. Table 4.1 shows nominal tempertures/heights that were obtained for the 18Z thresholds by adding the mean background brightness to the threshold values, and applying the temperatures obtained to the standard tropical atmosphere[9] to obtain heights. Certainly, some data contamination occurs with thin clouds and warm backgrounds making clouds appear lower, so actual cloud frequencies at a certain level may be somewhat higher than the ones reported in this study. It should also be noted that subtracting background brightness is necessary at the lower levels to prevent contamination from brighter/cooler surfaces (which could be misinterpreted to be low clouds), but when using higher thresholds to examine upper-level cloud frequencies, this background biasing could produce undesired discrimination against cloud frequencies which are above those brighter surfaces.

Table 4.1: Nominal heights and temperatures for 18Z climatology brightness thresholds.

Threshold Counts	Brightness Counts	Temperature °C	Height (meters)	Height (feet)
9	77	18.3	1611	5285
19	87	13.3	2537	8323
29	97	8.3	3463	11361
39	107	3.3	4389	14400
49	117	-1.7	5243	17201
59	127	-6.7	5957	19544
69	137	-11.7	6671	21886
79	147	-16.7	7386	24232
89	157	-21.7	8100	26575
99	167	-26.7	8814	28917
109	177	-32.2	9600	31496
119	187	-42.2	11029	36184
129	197	-52.2	12457	40869
139	207	-62.2	13886	45558

Though VOXEL excelled at displaying these 3-D data sets, the octree data compression was not as high as originally hoped. Both data sets began with 513 memory blocks and after encoding, the 18Z octree required 431 blocks and the 15Z octree needed 416 blocks. While

[9]As defined by Indian meteorologist Pisharoty in 1959

the savings are between 15 and 20%, this is hardly a quantum leap in efficiency. Better compression factors could have been attained with more coherent data.

4.2.3 SHOWING DIFFERENCES BETWEEN CLIMATE DATA SETS

Weather forecasters find it valuable to know the climatology of an area to make more accurate forecasts. Just as valuable is the knowledge of the diurnal changes in the climatology. By merging data sets from two valid times, the changes between them become readily apparent.

An example of merged data sets is shown in Figure 4.32 in which 15Z and 18Z IR climatologies are displayed in white and orange respectively (yellow is their intersection). In this figure, the height coordinate represents cloud frequency-of-occurrence. To merge the data GOESVOX was programmed to set all the 15Z data to a color table intensity of 10 (white). Then 65 was added to every array element in which 18Z data occurred. This left color values of 65 (orange) in array elements that were not occupied by 15Z data, and values of 75 (yellow) in elements that were occupied by both data sets. Boundaries were added through the use of the "wire frame" graphic overlay from Figure 4.4 (though reset to blue for better contrast).

This type of VOXEL display makes the differences graphically stand out. It is clear that clouds occurred much more frequently over the land masses at 18Z than at 15Z (for a 9 count brightness threshold). It would seem that land/sea breezes played a role in this pattern, since 15Z clouds were generally more frequent just off the coast where direct circulation would cause greater subsidence by 18Z.

The octree encoding for this display was much more efficient than the last application, as evidenced by the large yellow blocks seen on the side of the octree. Though the data was still somewhat incoherent, the encoding process reduced the required memory from 513 blocks to 177 blocks: a 65% savings.

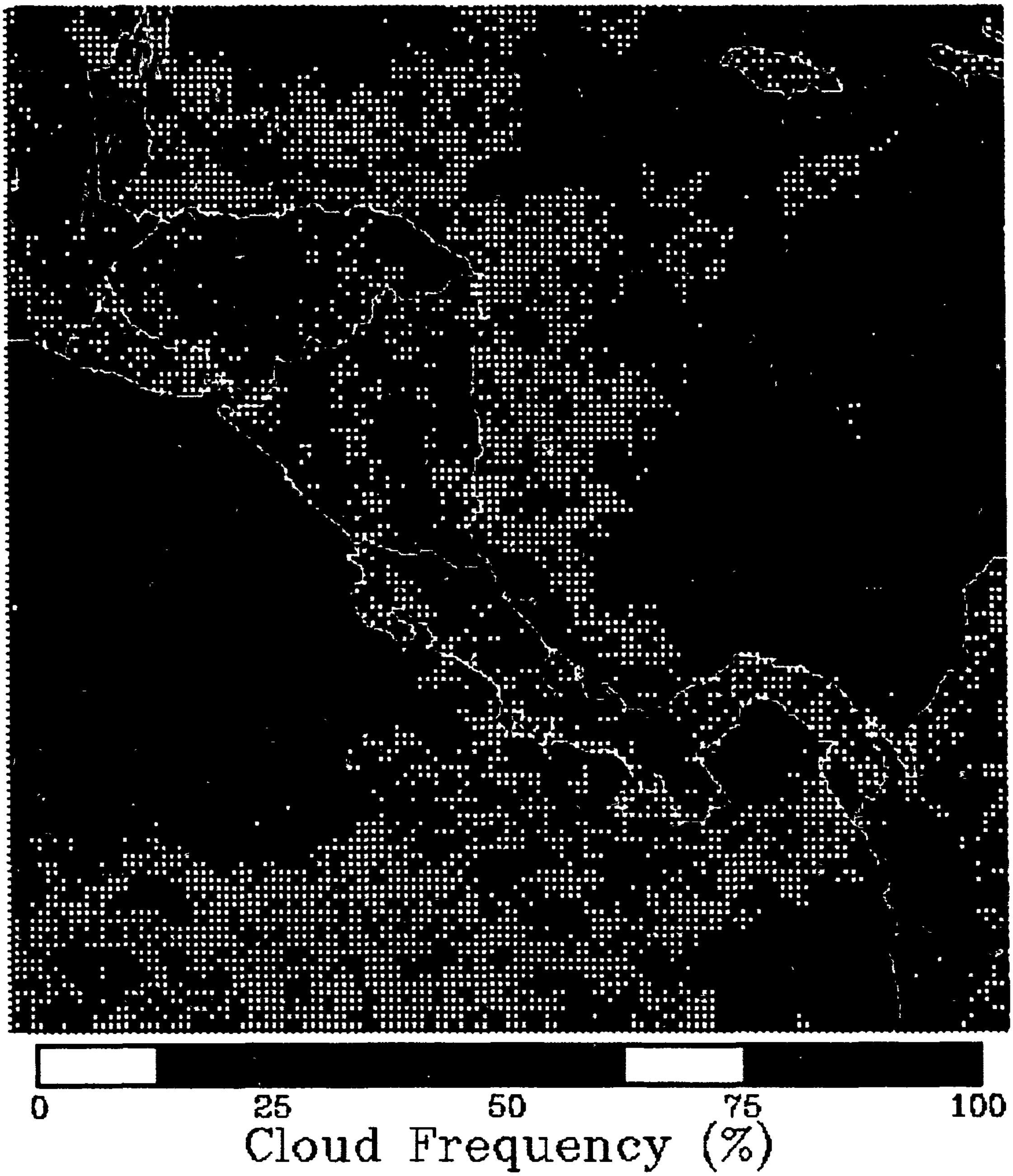

Figure 4.5: 18Z December 1990 IR cloud climatology with brightness threshold set 9 counts above background brightness.

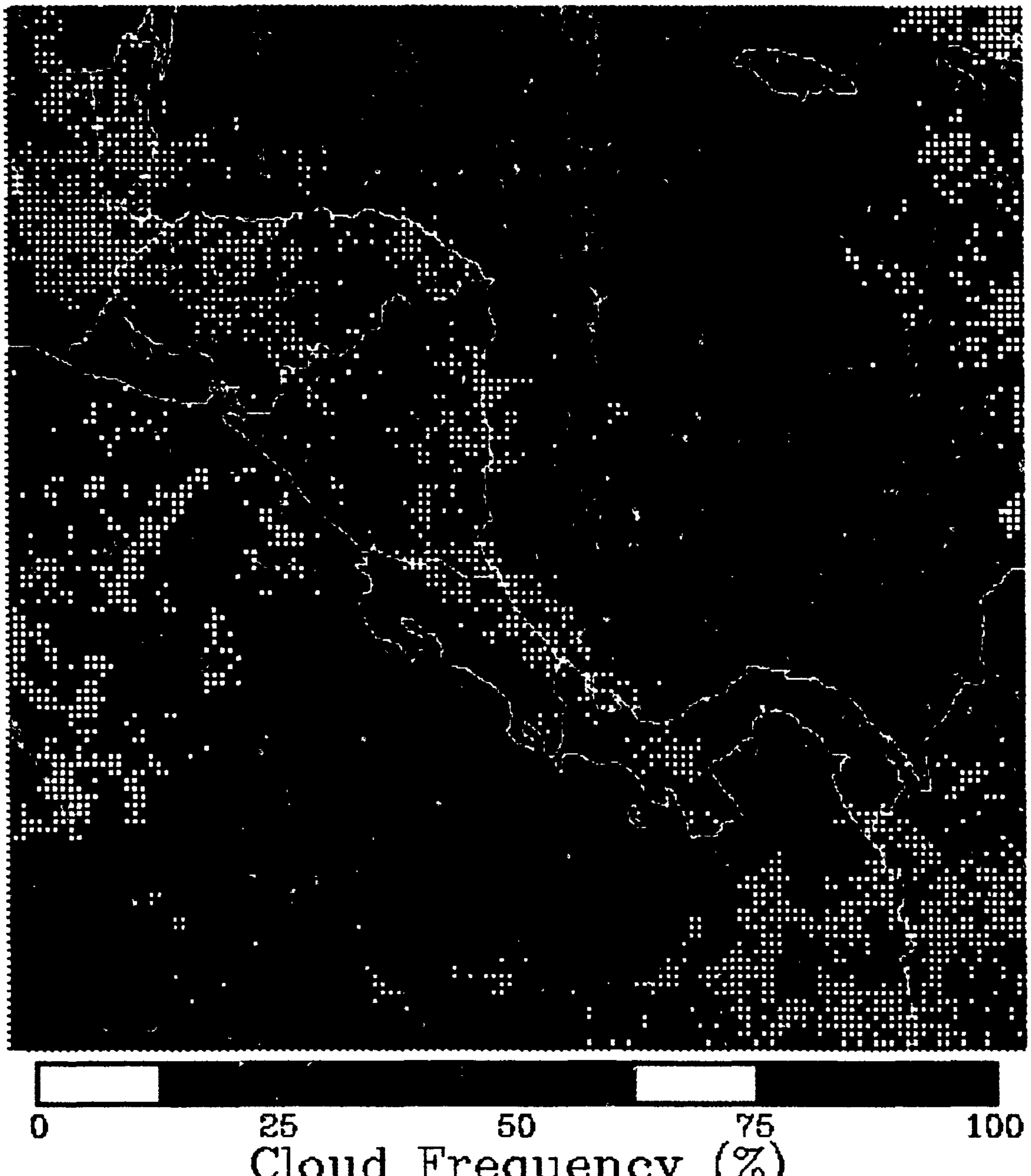

Figure 4.6: 18Z December 1990 IR cloud climatology with brightness threshold set 19 counts above background brightness.

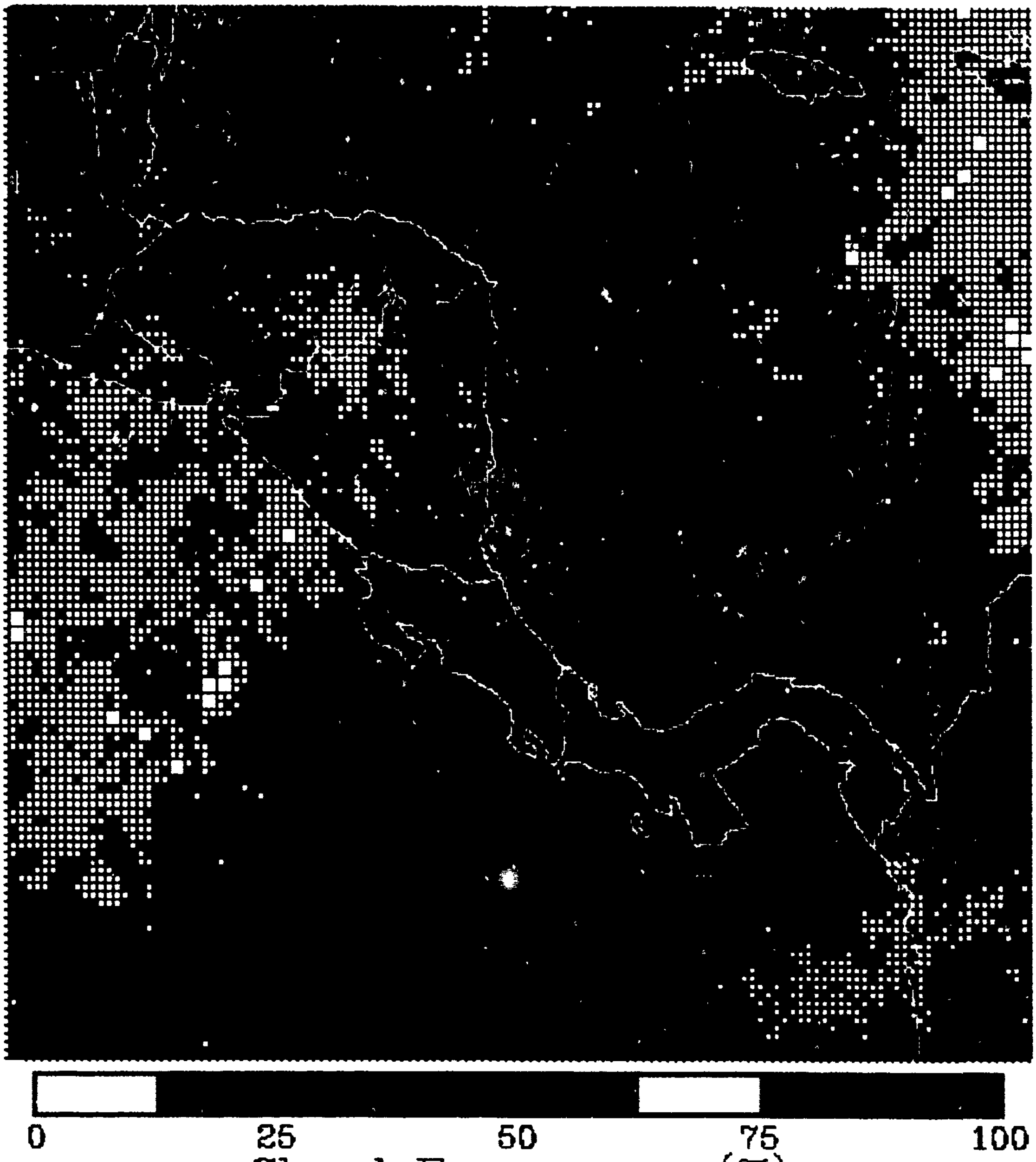

Figure 4.7: 18Z December 1990 IR cloud climatology with brightness threshold set 29 counts above background brightness.

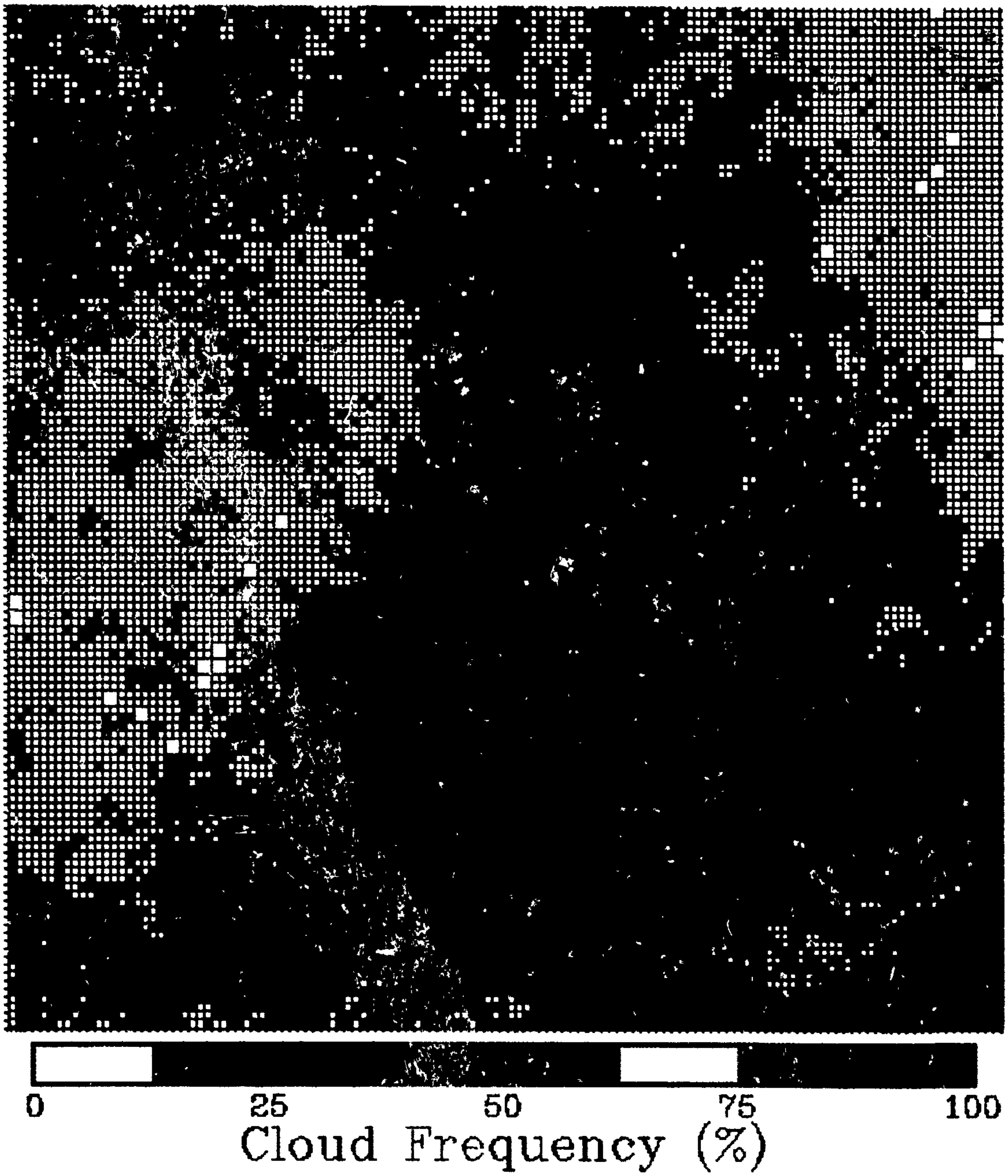

Figure 4.8: 18Z December 1990 IR cloud climatology with brightness threshold set 39 counts above background brightness.

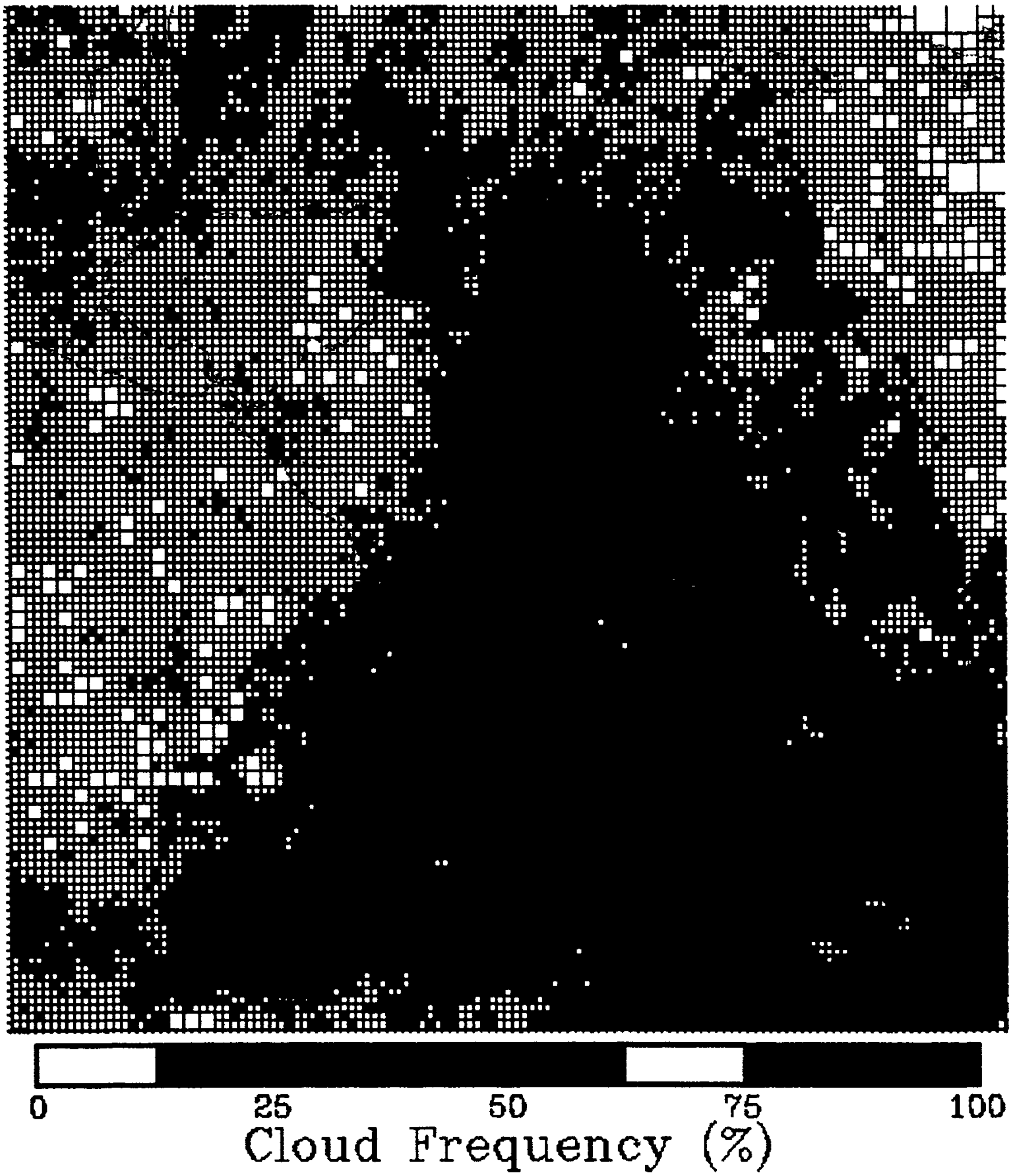

Figure 4.9: 18Z December 1990 IR cloud climatology with brightness threshold set 49 counts above background brightness.

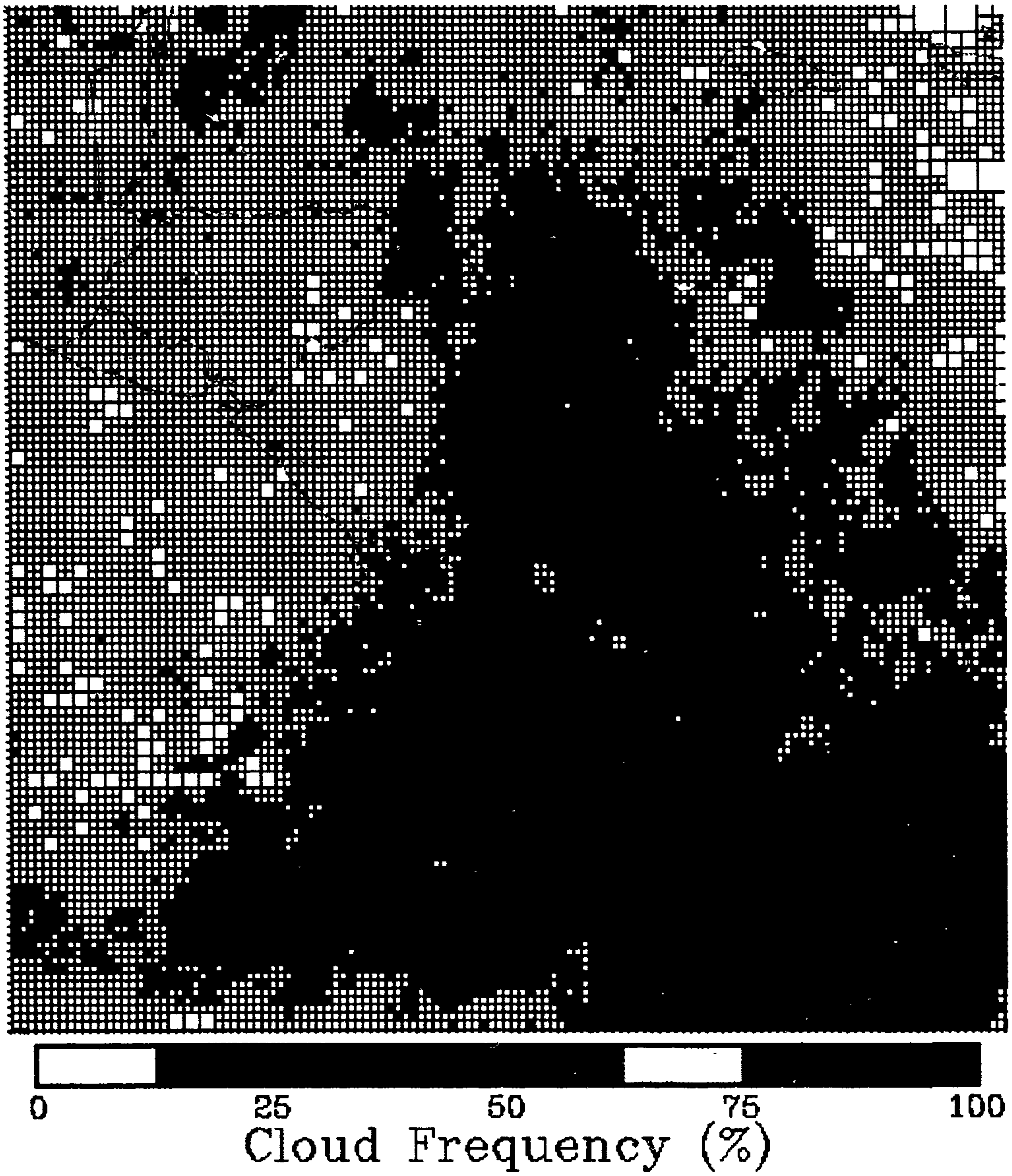

Figure 4.10: 18Z December 1990 IR cloud climatology with brightness threshold set 59 counts above background brightness.

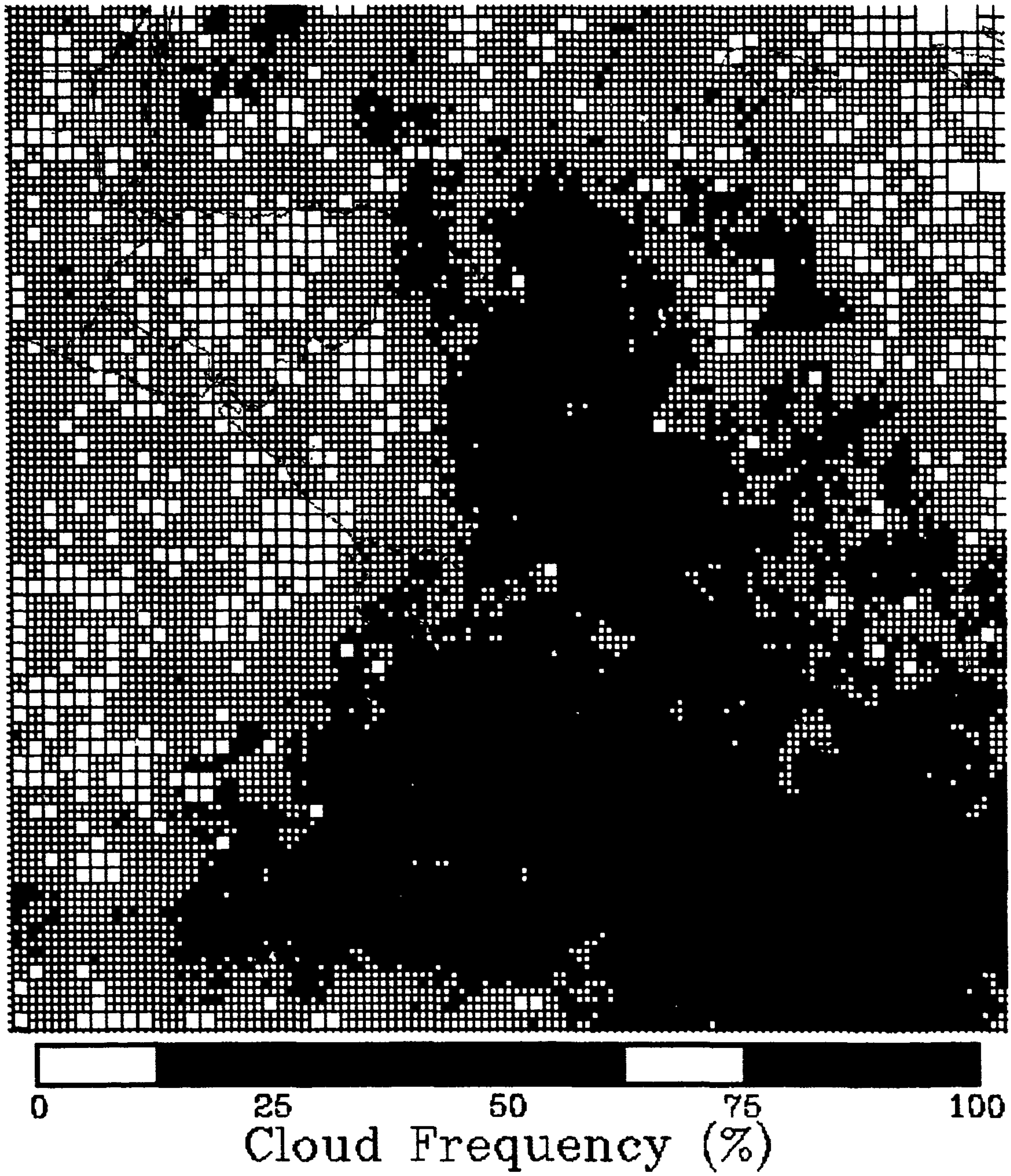

Figure 4.11: 18Z December 1990 IR cloud clir.:atology with brightness threshold set 69 counts above background brightness.

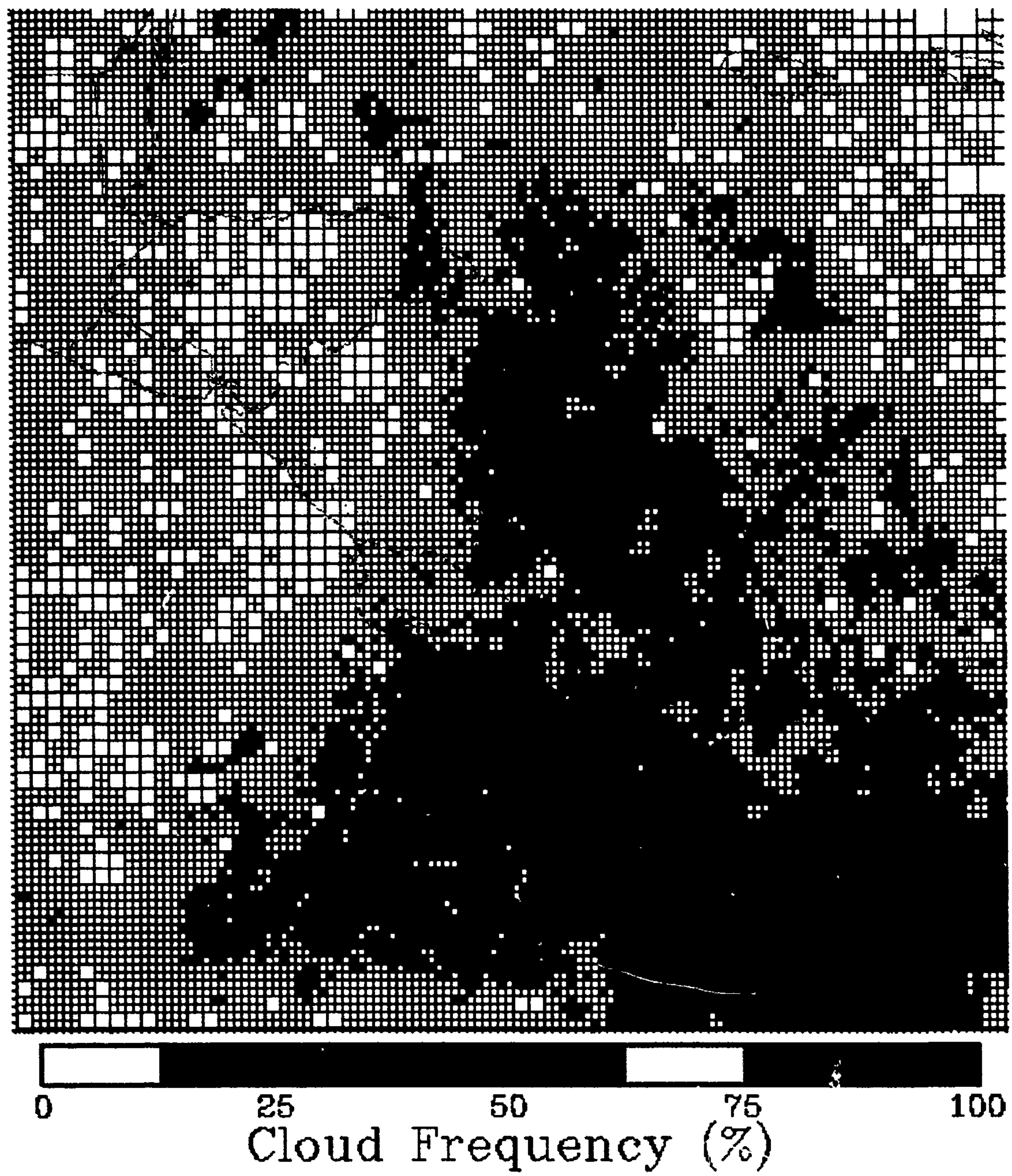

Figure 4.12: 18Z December 1990 IR cloud climatology with brightness threshold set 79 counts above background brightness.

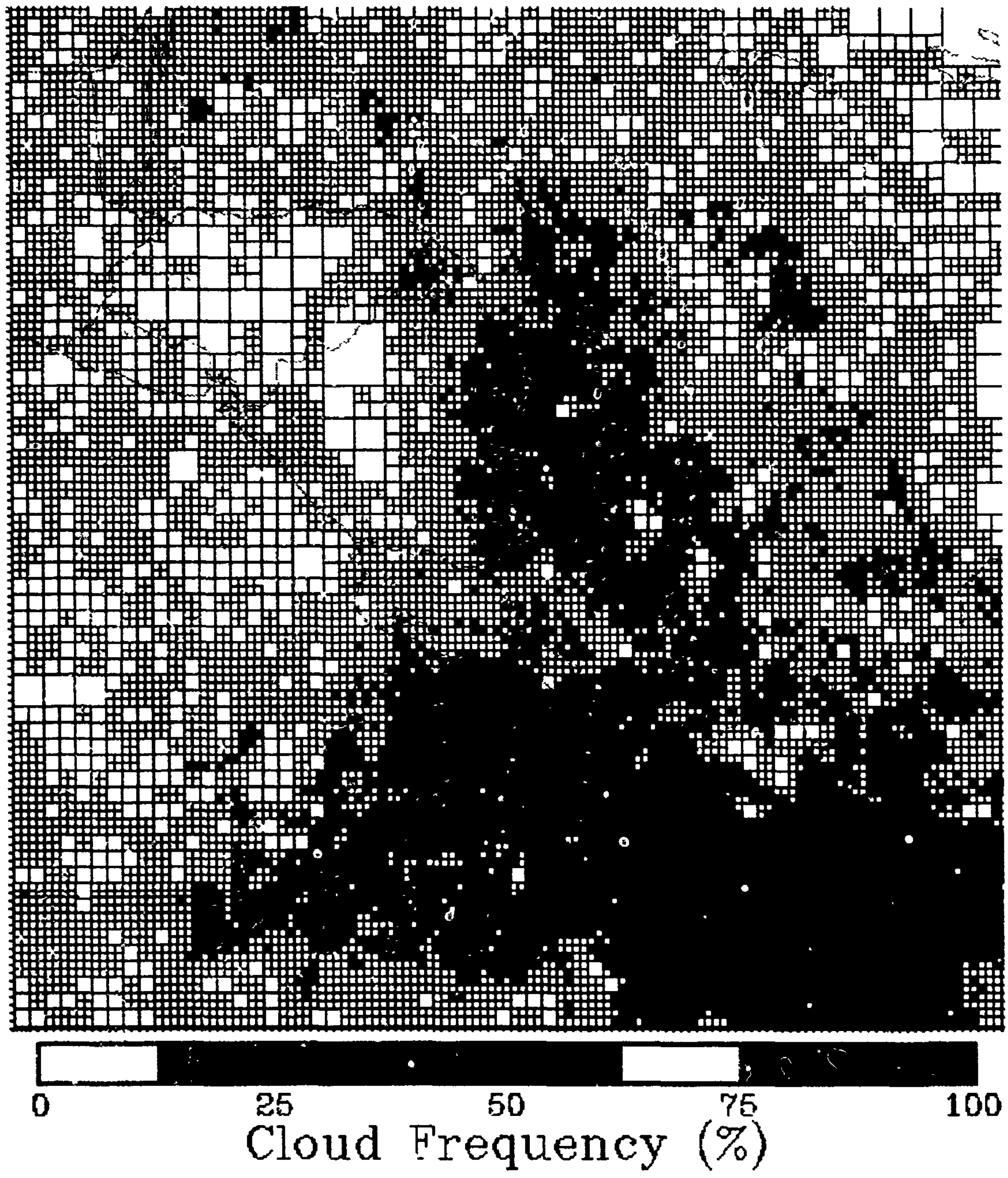

Figure 4.13: 18Z December 1990 IR cloud climatology with brightness threshold set 89 counts above background brightness.

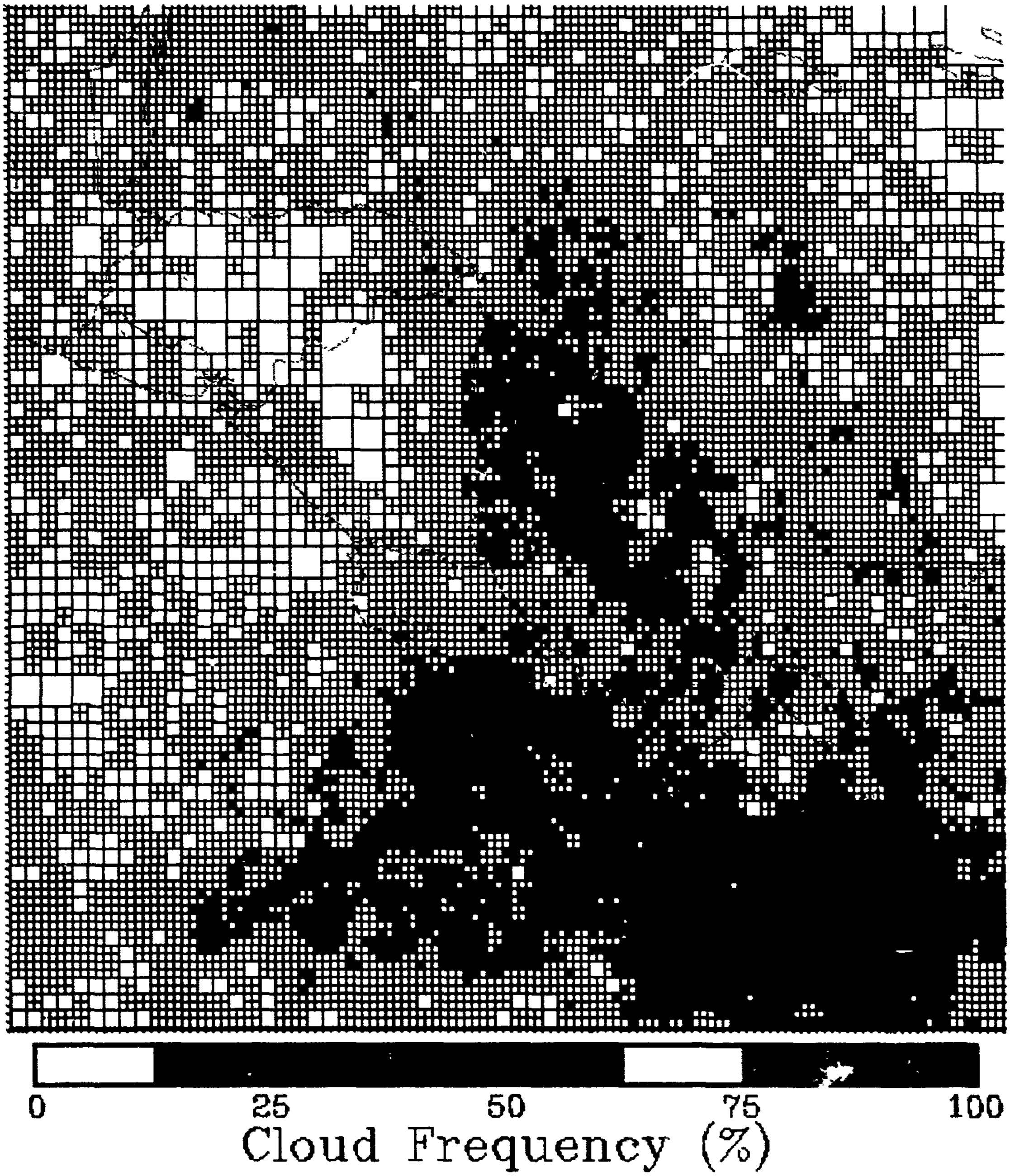

Figure 4 14: 18Z December 1990 IR cloud climatology with brightness threshold set 99 counts above background brightness.

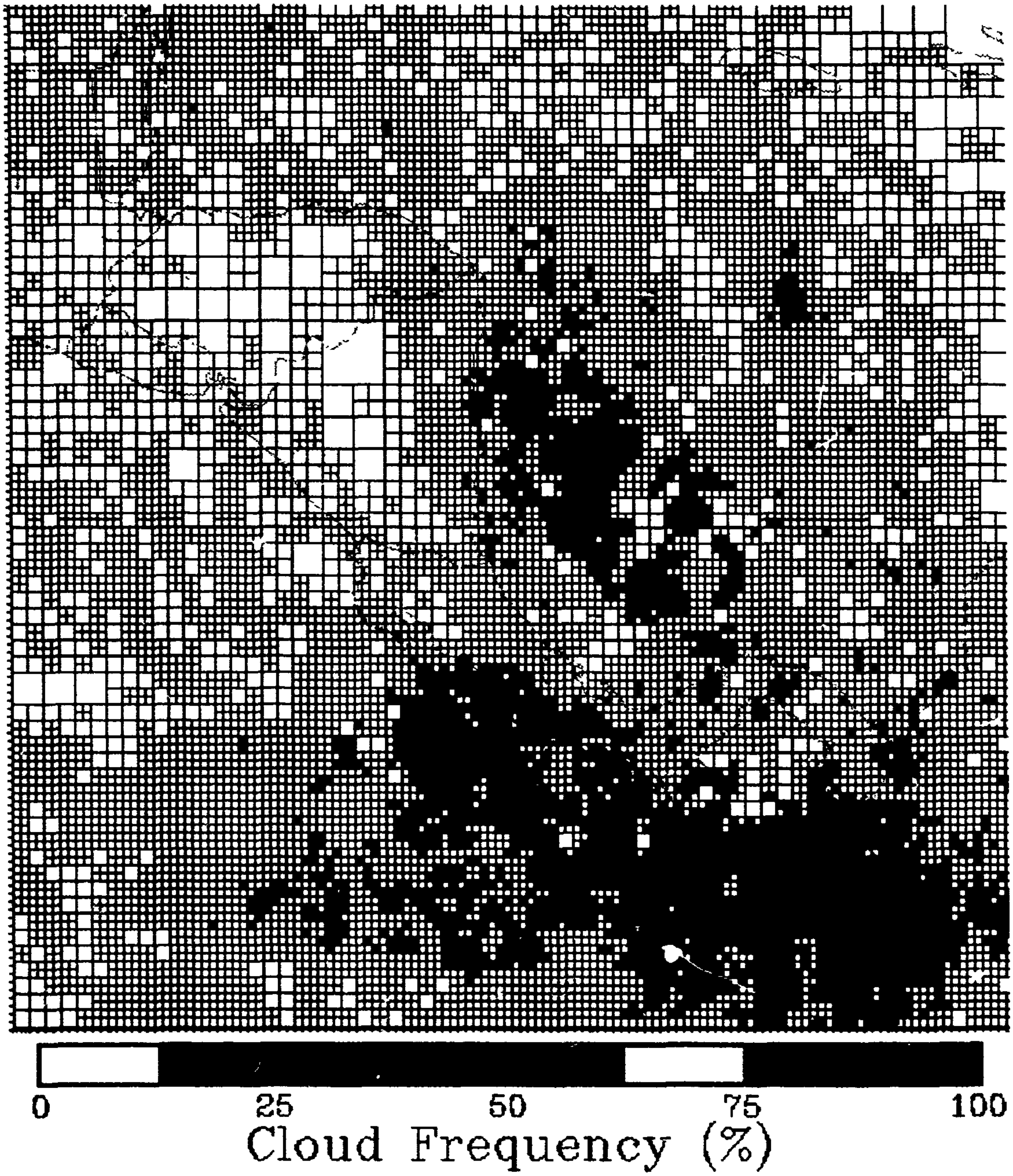

Figure 4.15: 18Z December 1990 IR cloud climatology with brightness threshold set 109 counts above background brightness.

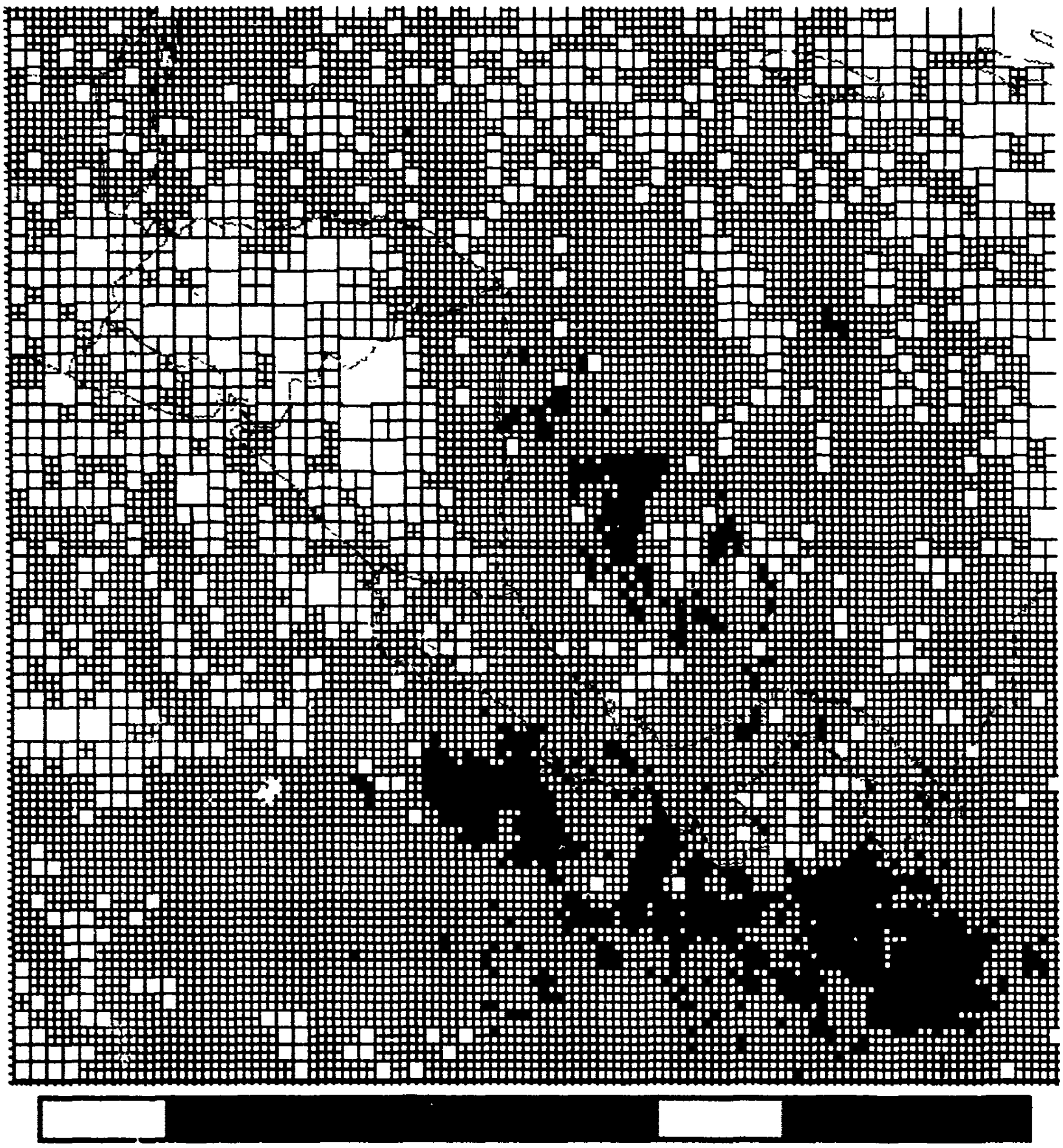

Figure 4.16: 18Z December 1990 IR cloud climatology with brightness threshold set 119 counts above background brightness

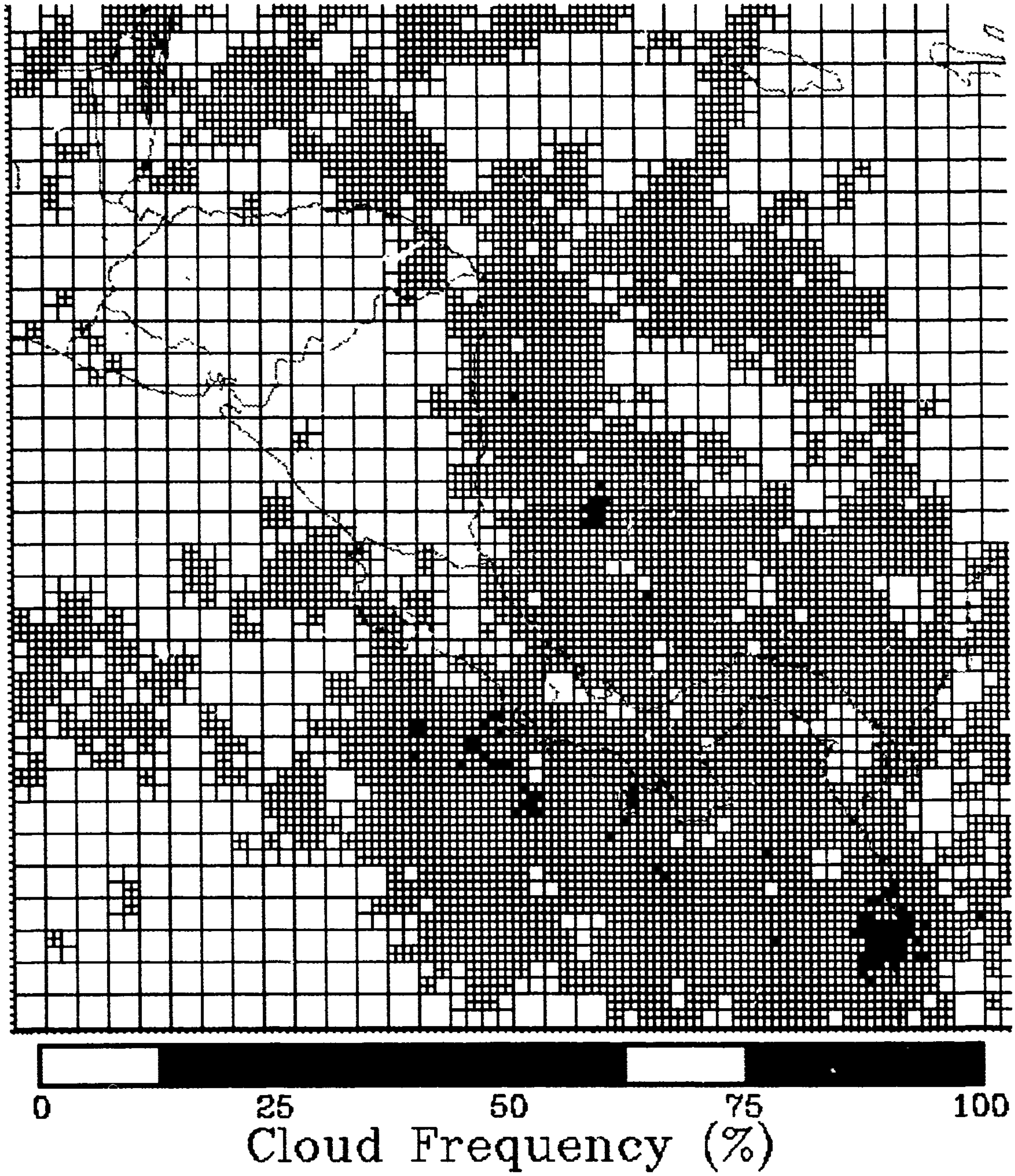

Figure 4.17: 18Z December 1990 IR cloud climatology with brightness threshold set 129 counts above background brightness.

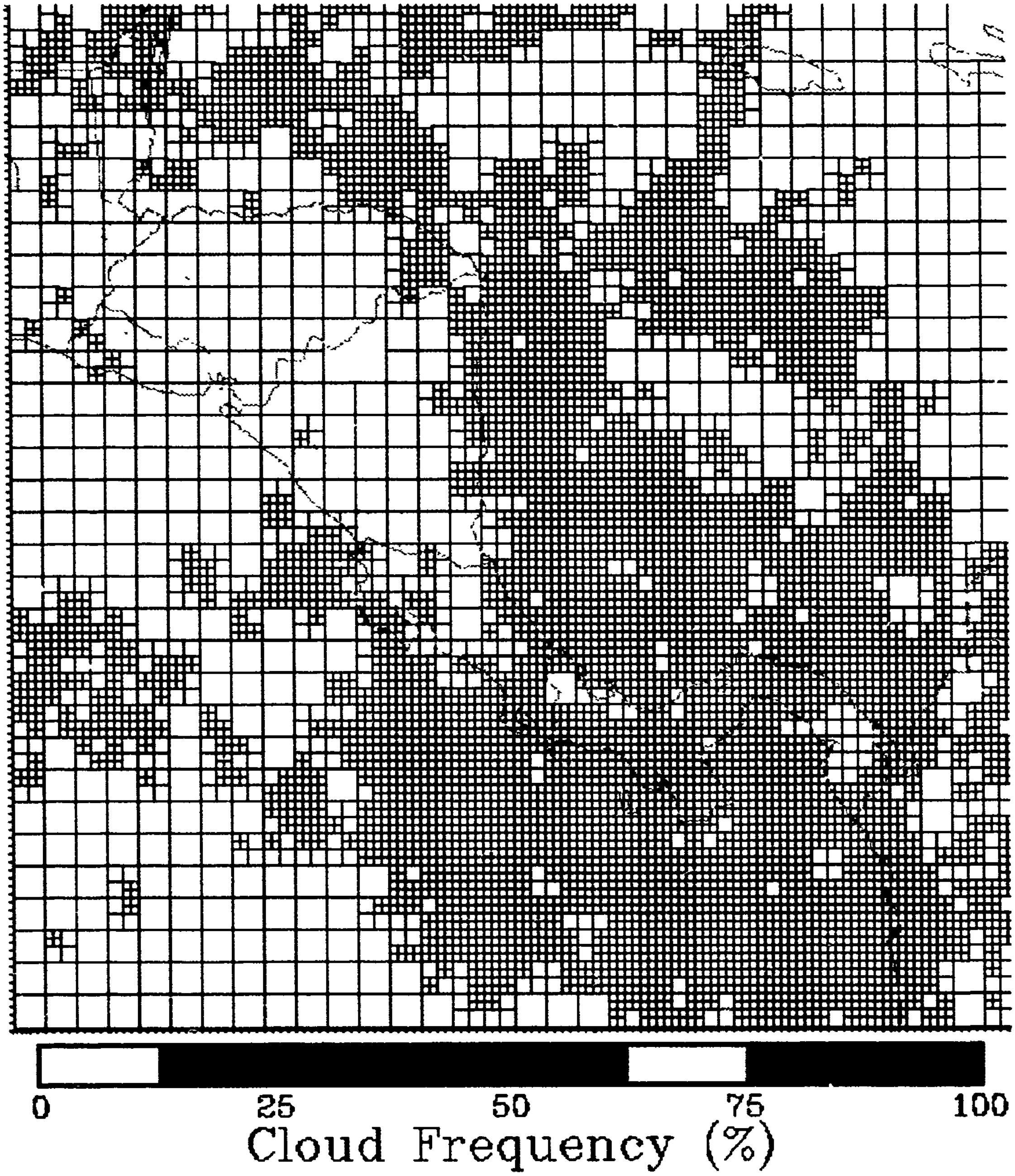

Figure 4.18: 18Z December 1990 IR cloud climatology with brightness threshold set 139 counts above background brightness.

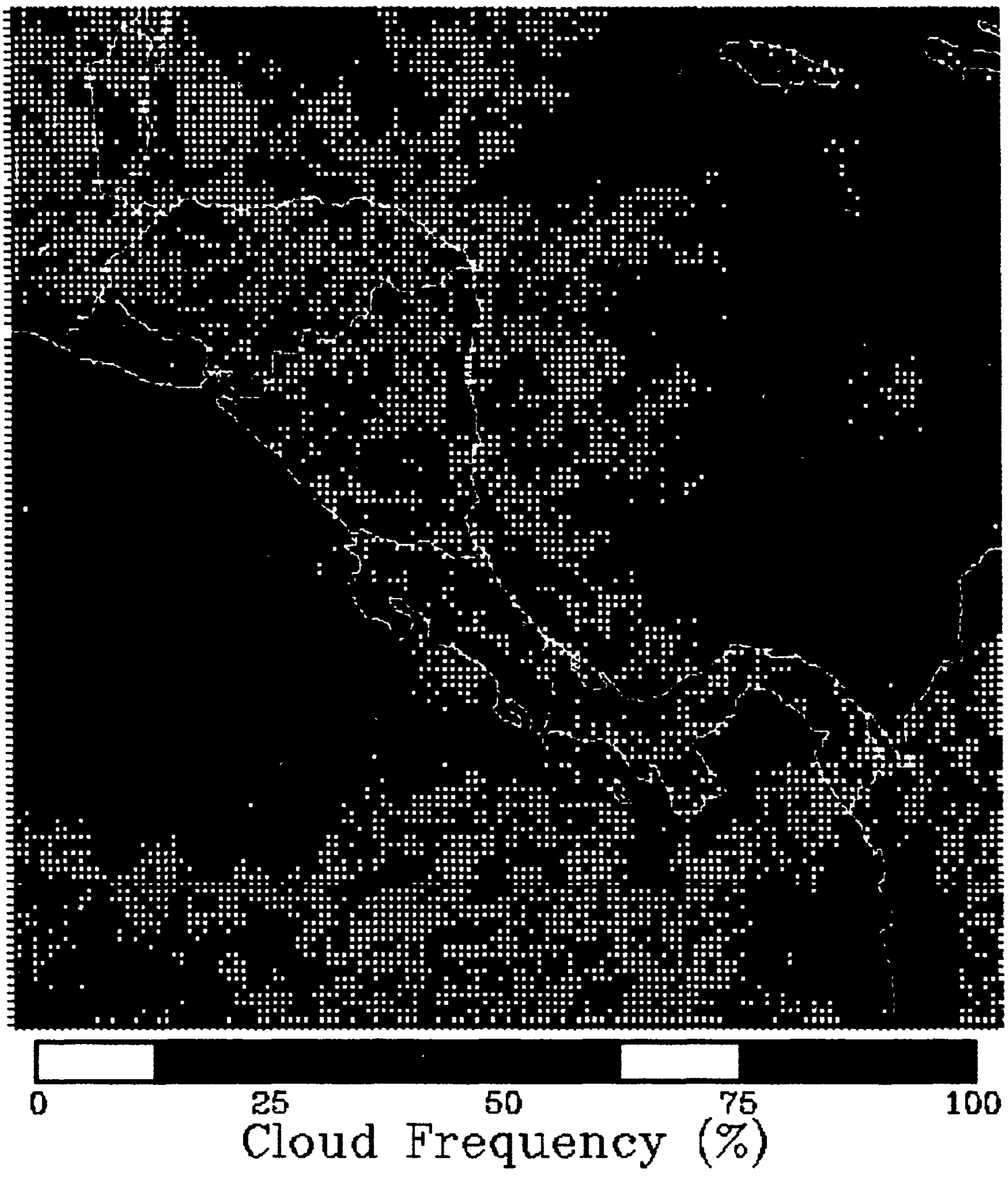

Figure 4.19: 15Z December 1990 IR cloud climatology with brightness threshold set 9 counts above background brightness.

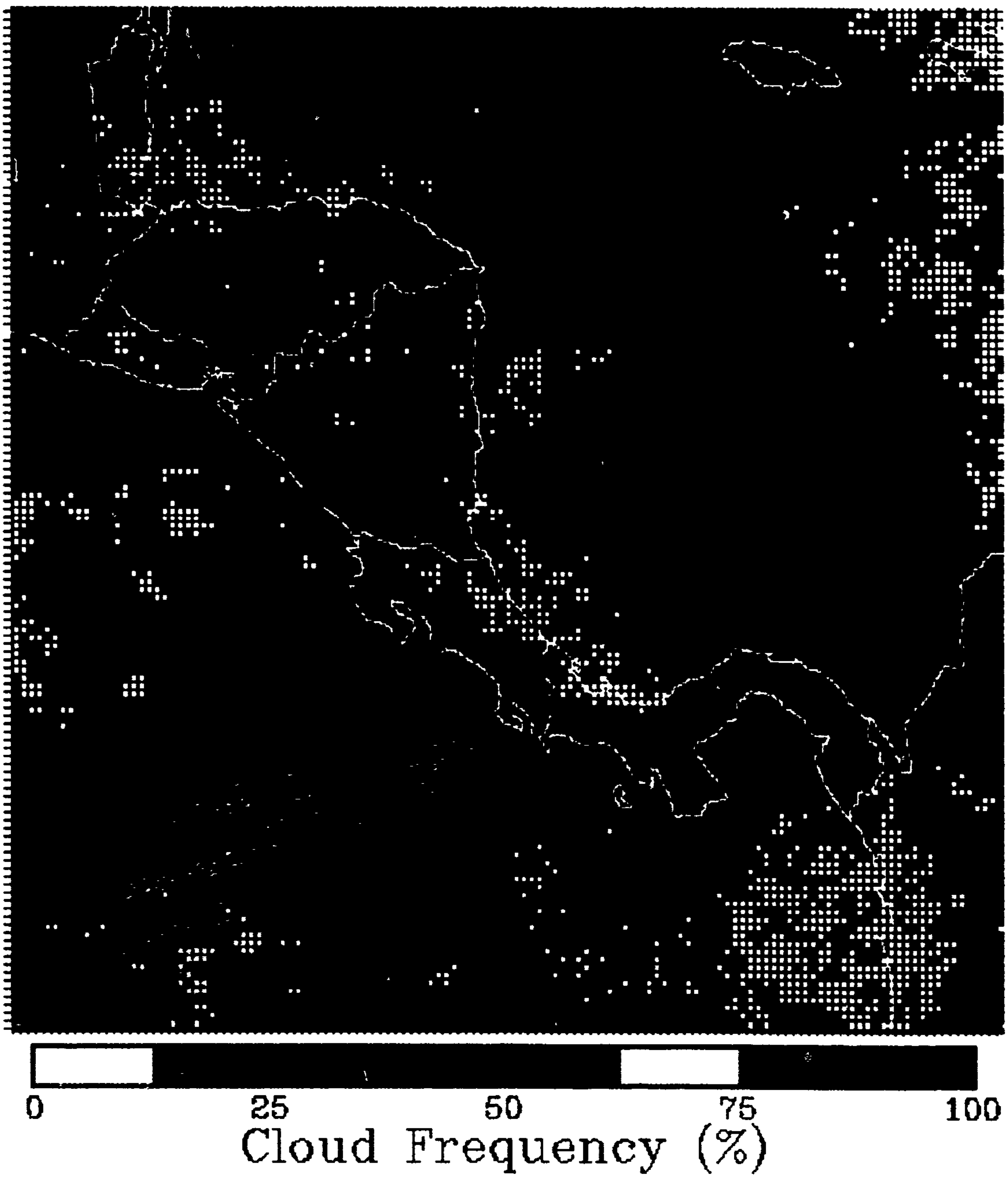

Figure 4.20: 15Z December 1990 IR cloud climatology with brightness threshold set 19 counts above background brightness.

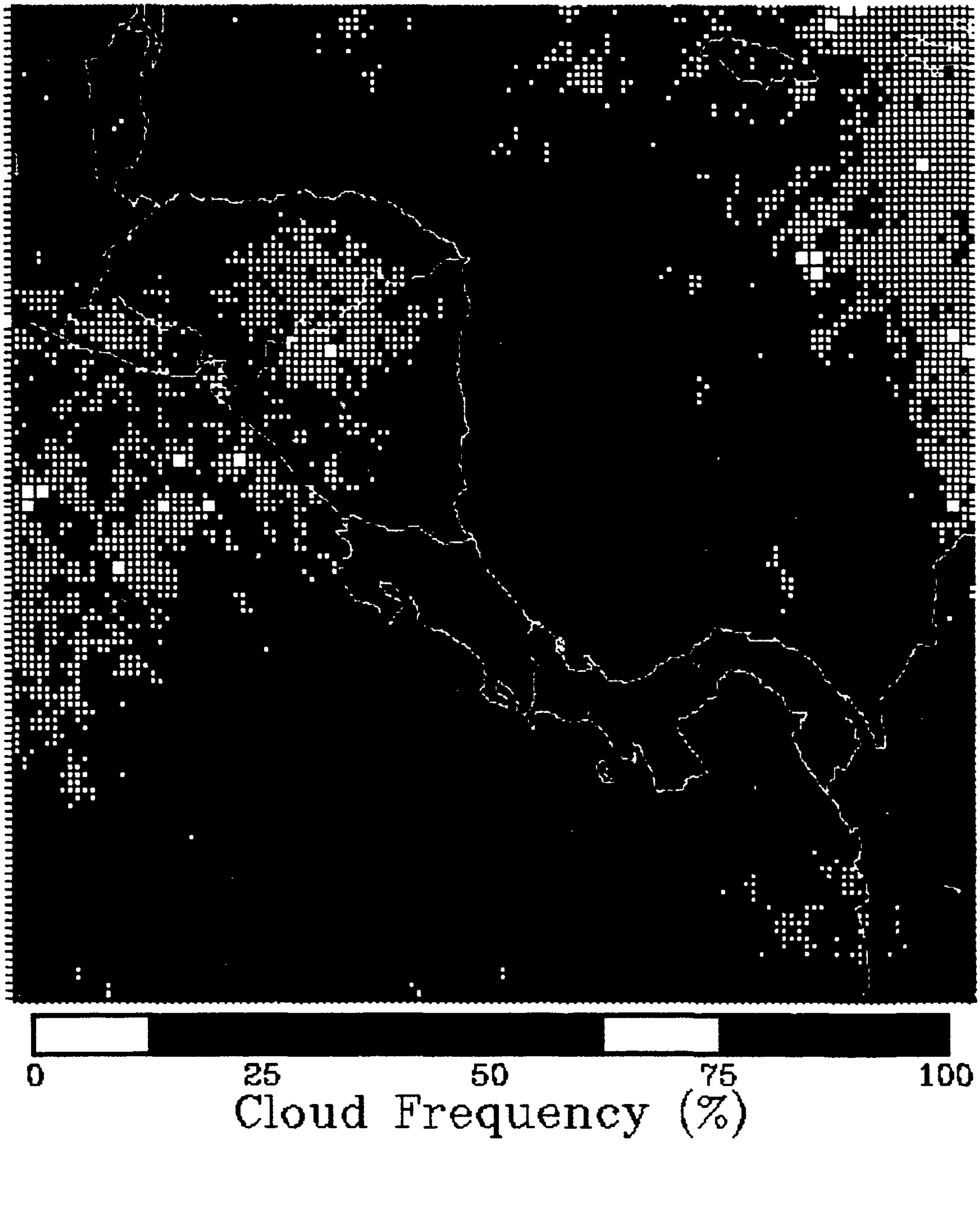

Figure 4.21: 15Z December 1990 IR cloud climatology with brightness threshold set 29 counts above background brightness.

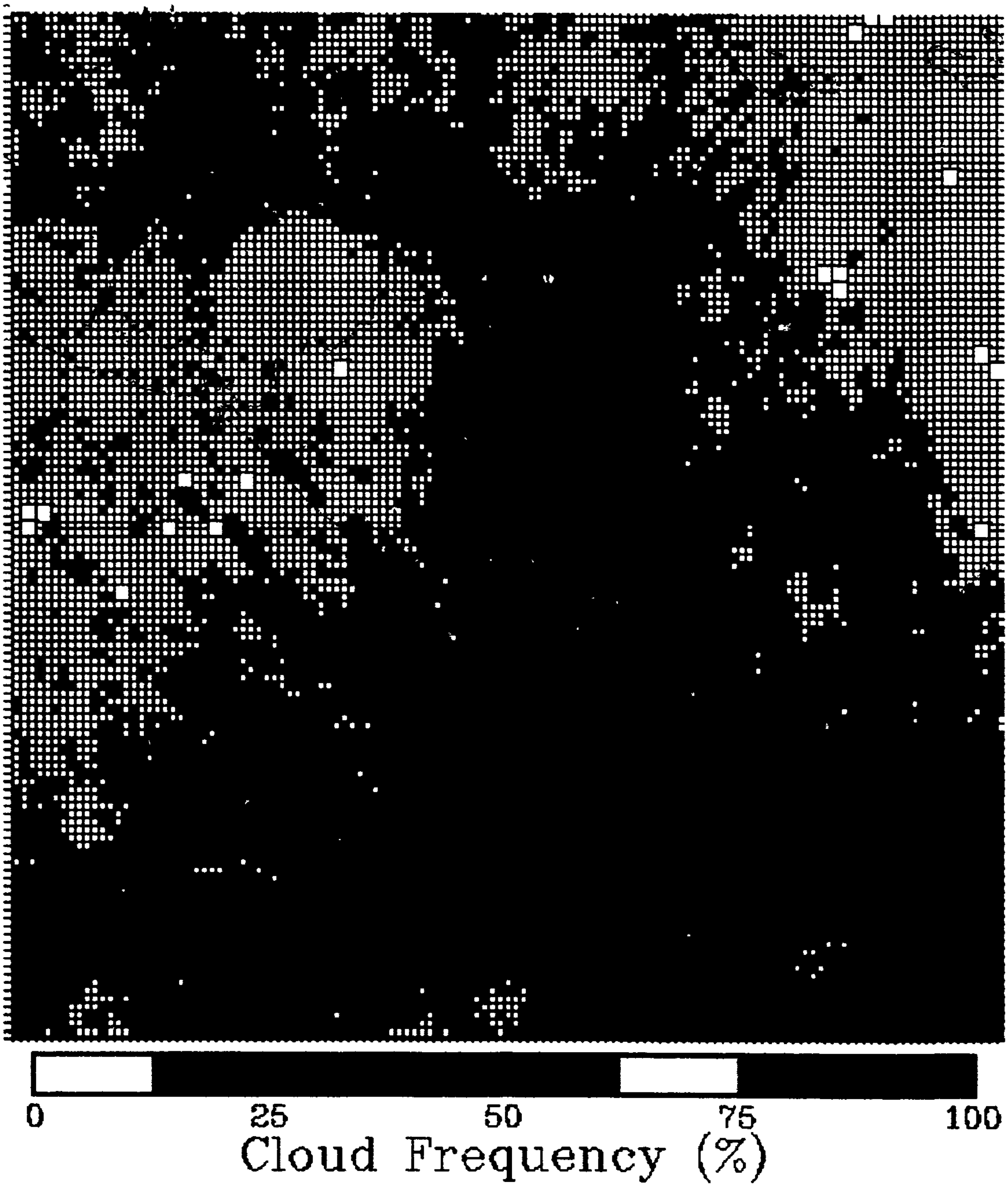

Figure 4 22: 15Z December 1990 IR cloud climatology with brightness threshold set 39 counts above background brightness

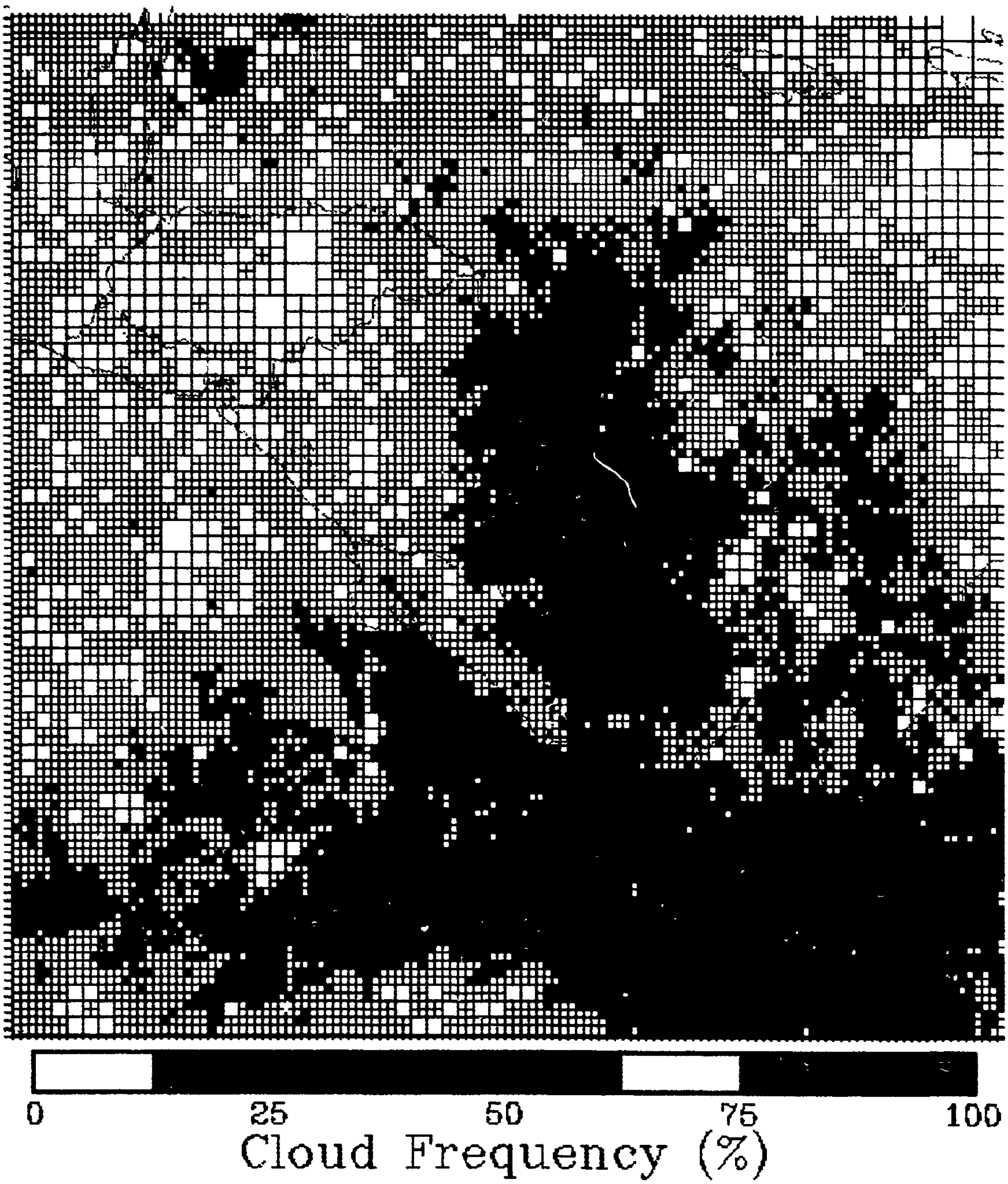

Figure 4.23: 15Z December 1990 IR cloud climatology with brightness threshold set 69 counts above background brightness.

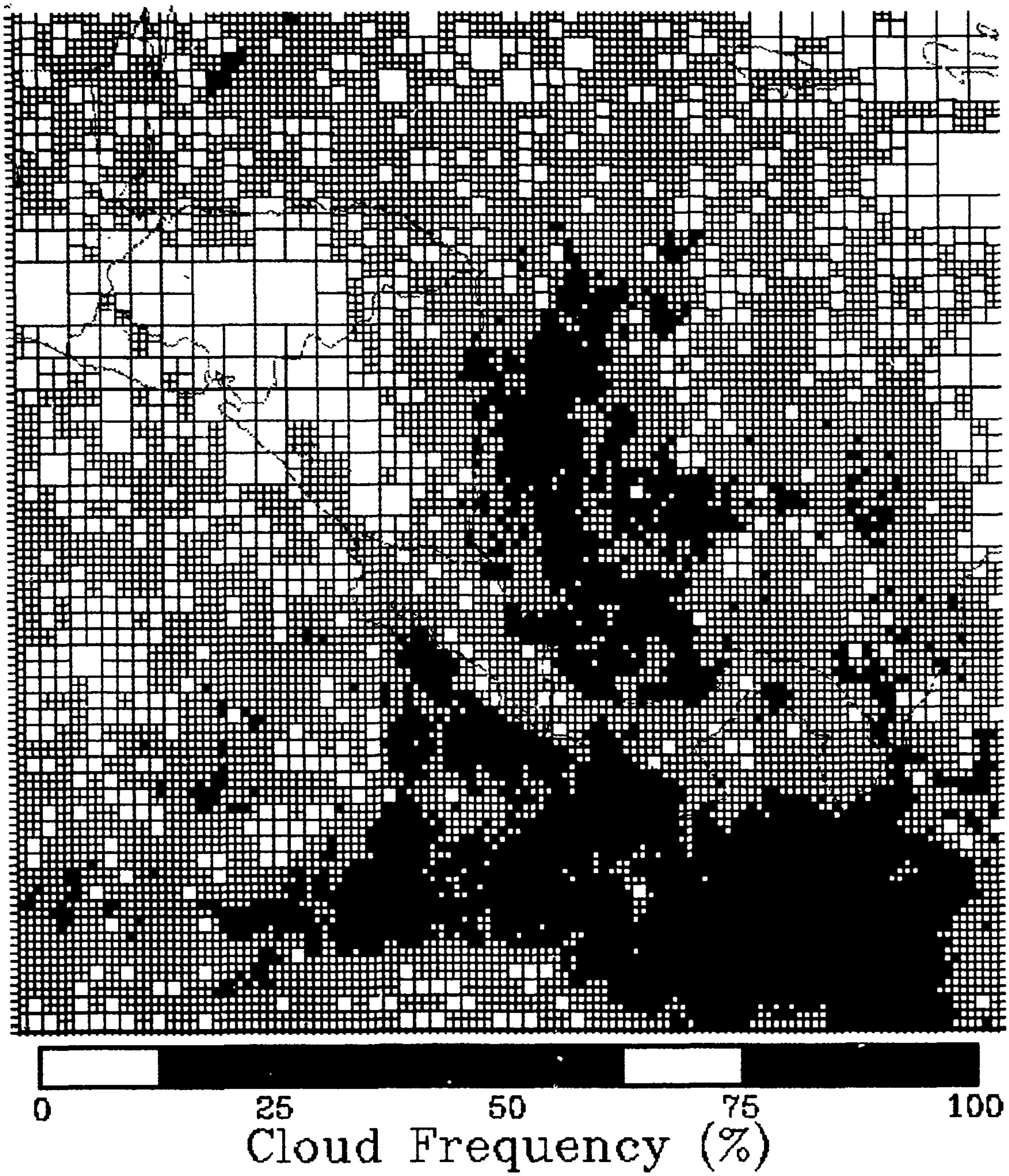

Figure 4.24: 15Z December 1990 IR cloud climatology with brightness threshold set 99 counts above background brightness.

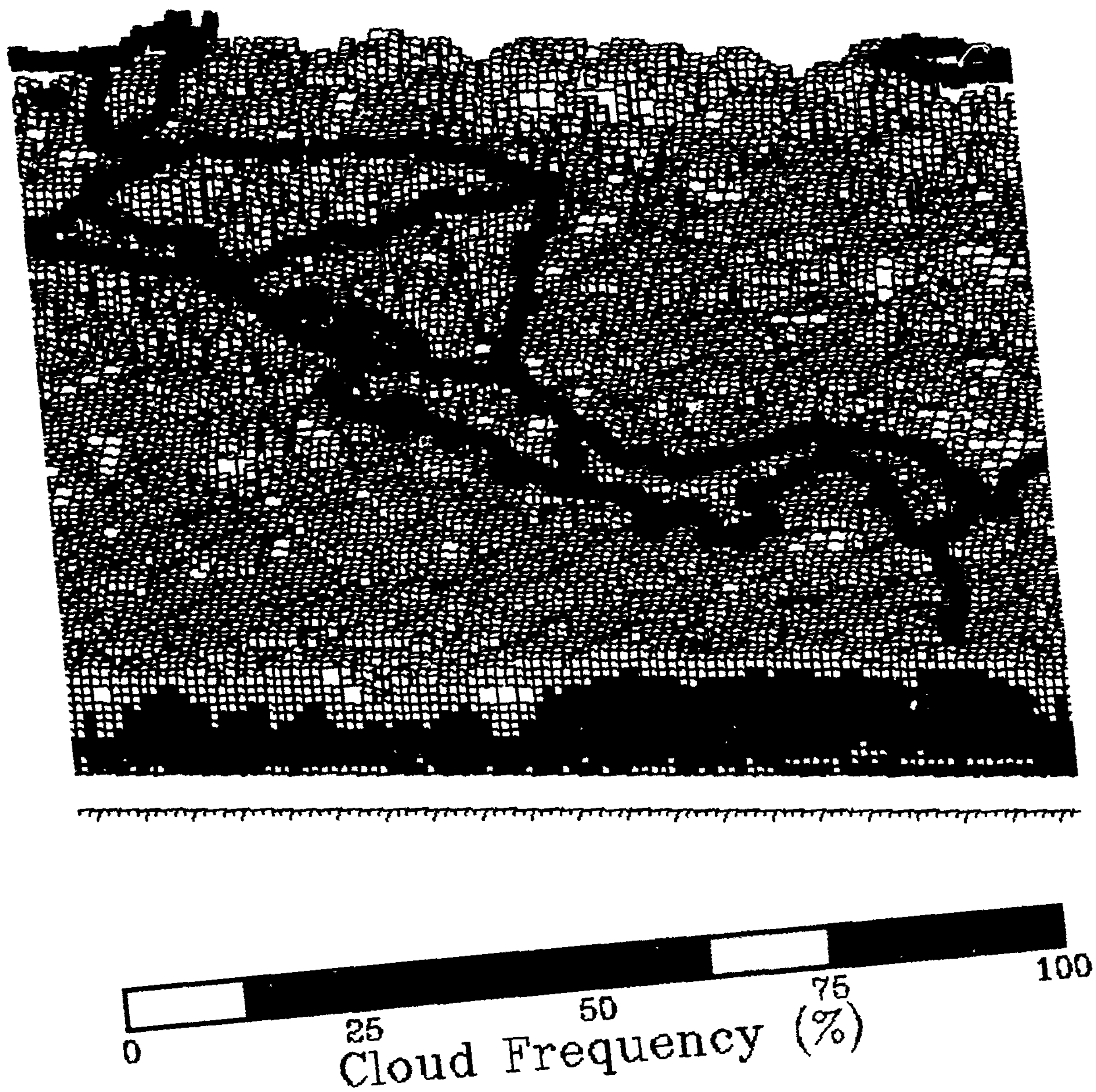

Figure 4.25: 15Z December 1990 3-D IR cloud climatology with vertical cross-section through the ITCZ.

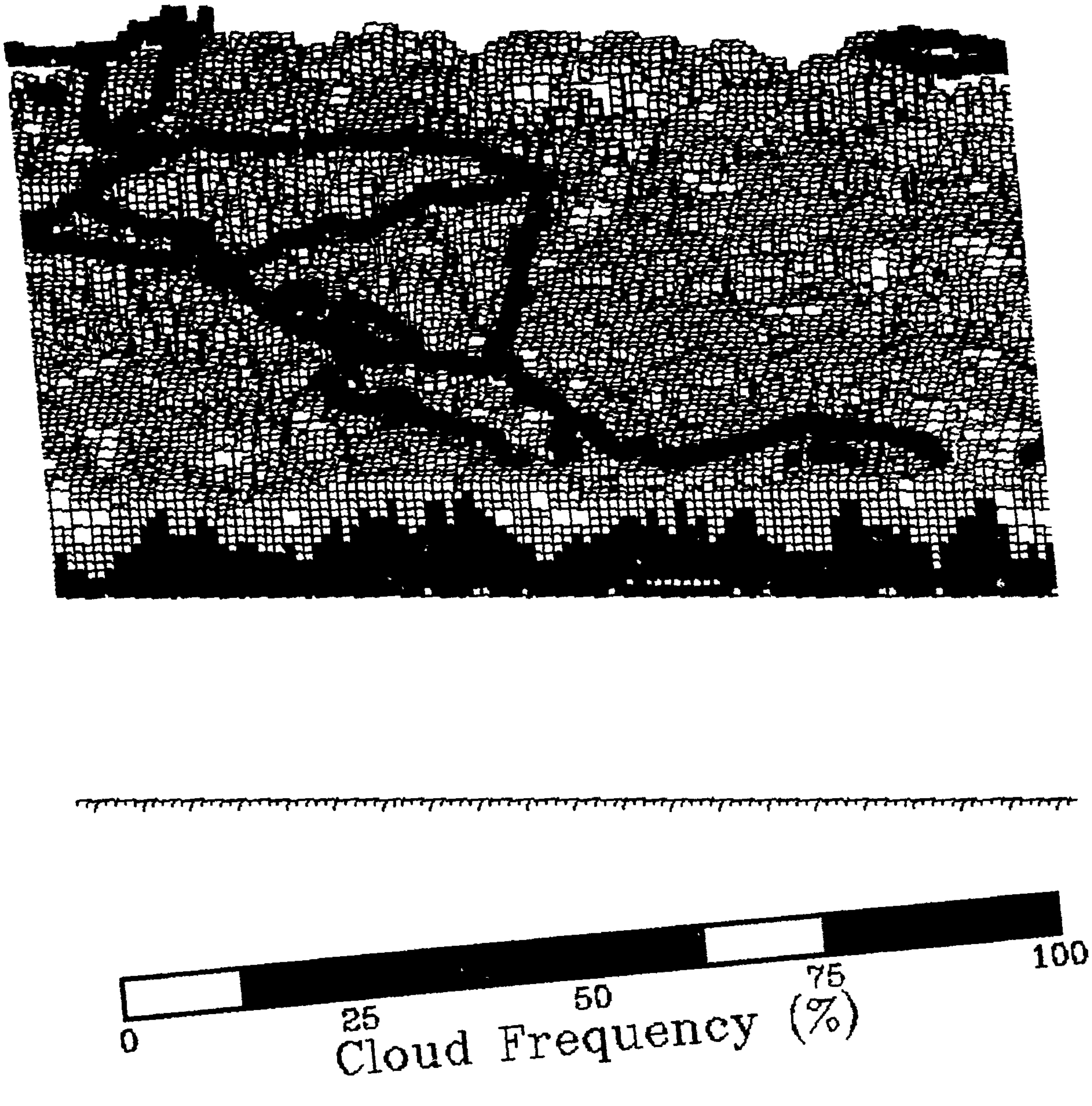

Figure 4.26: 15Z December 1990 3-D IR cloud climatology with vertical cross-section through Panama.

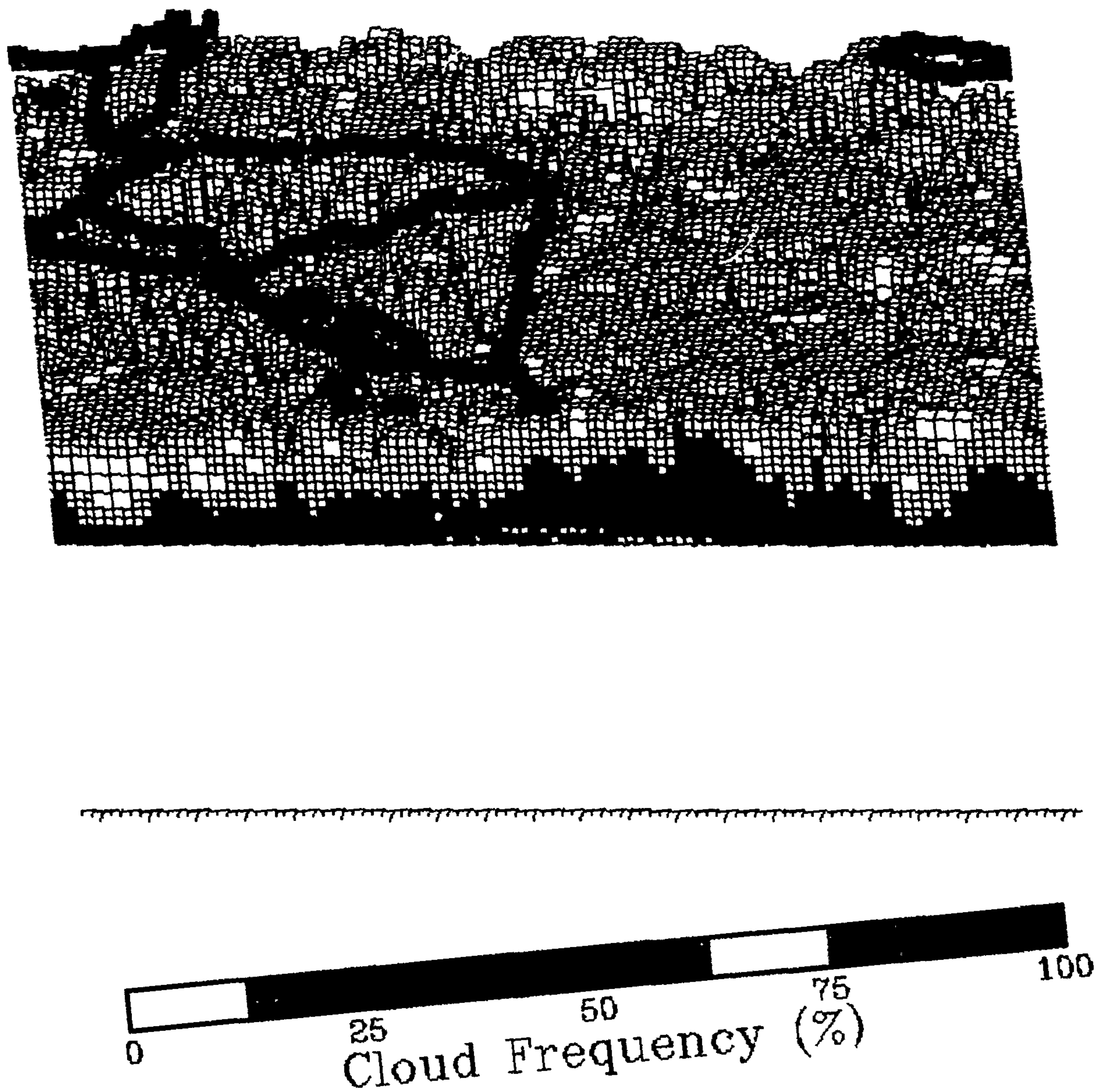

Figure 4.27: 15Z December 1990 3-D IR cloud climatology with vertical cross-section through Costa Rica.

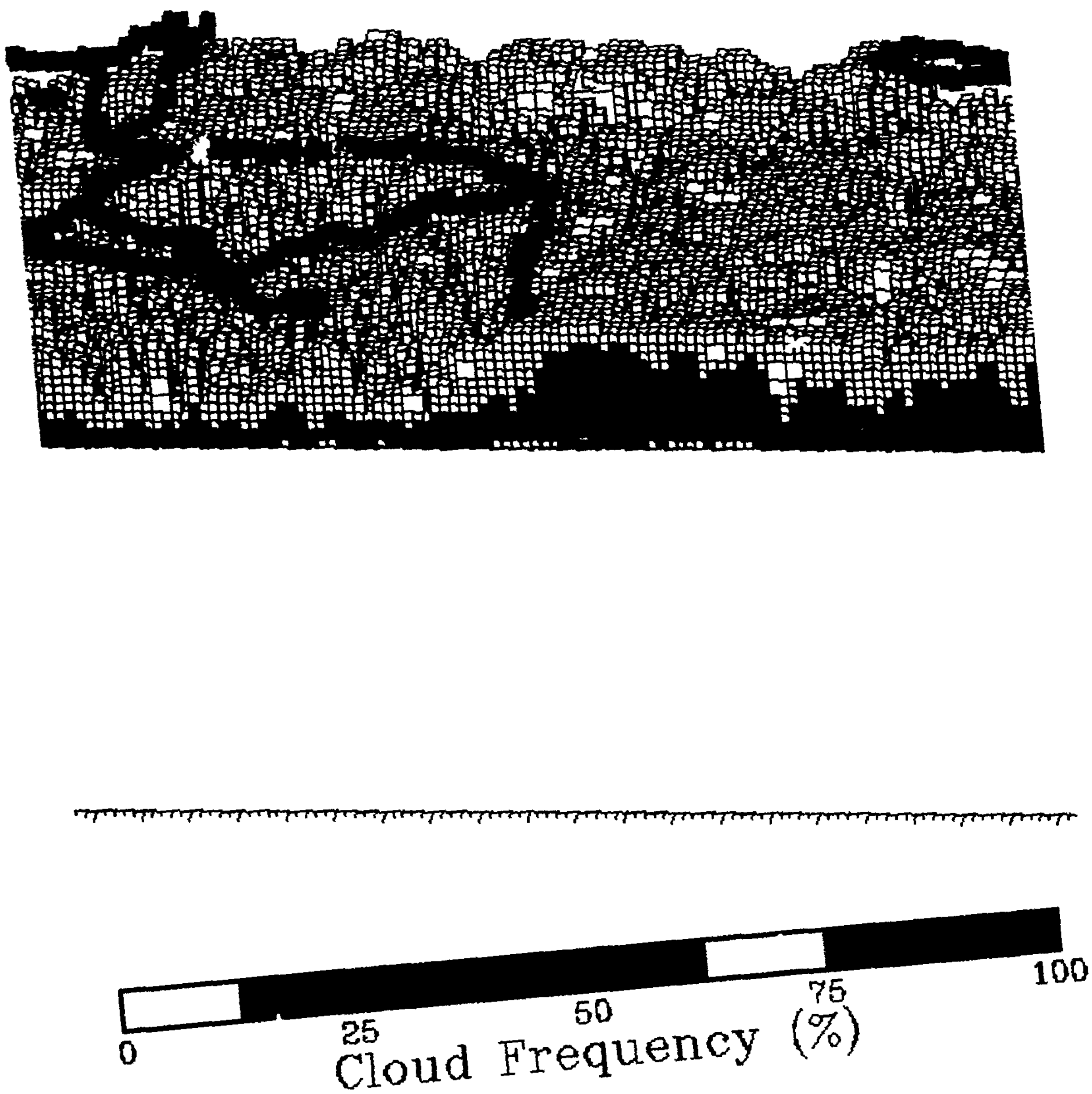

Figure 4.25: 15Z December 1990 3-D IR cloud climatology with vertical cross-section through Nicaragua.

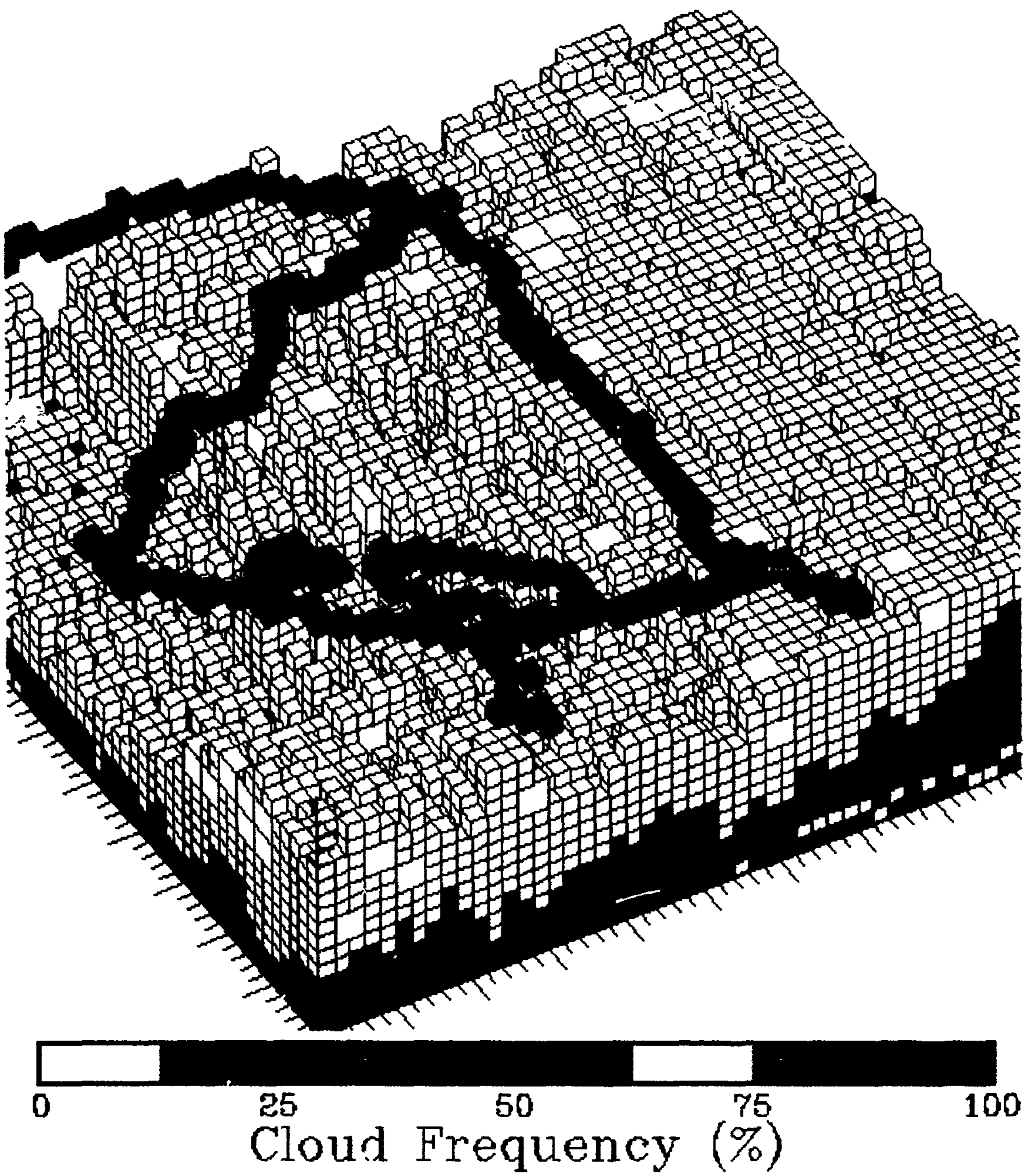

Figure 4.29: 15Z December 1990 3-D IR cloud climatology of Nicaraguan region at 1% cloud threshold

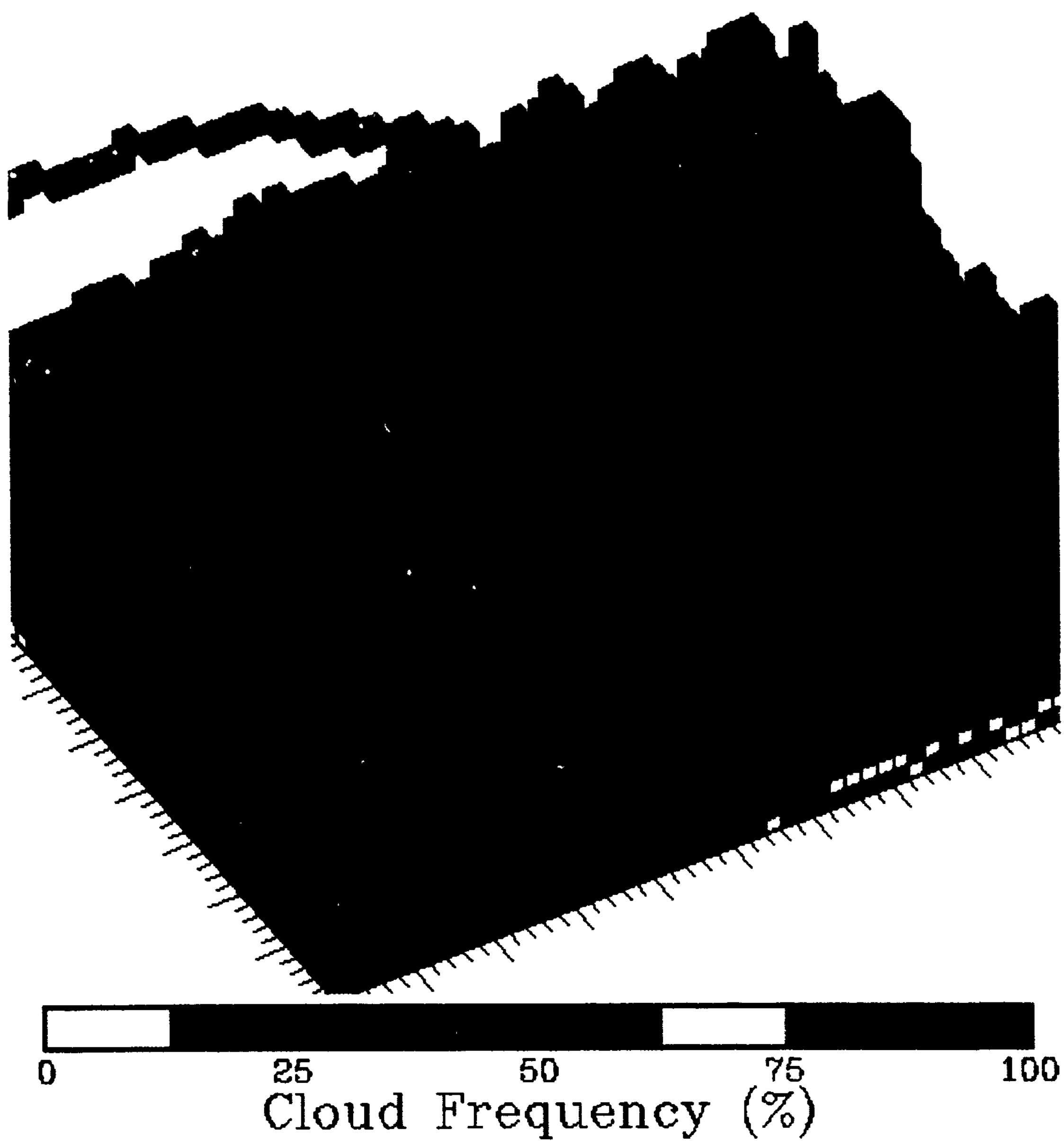

Figure 4.30: 15Z December 1990 3-D IR cloud climatology of Nicaraguan region at 12.5% cloud threshold.

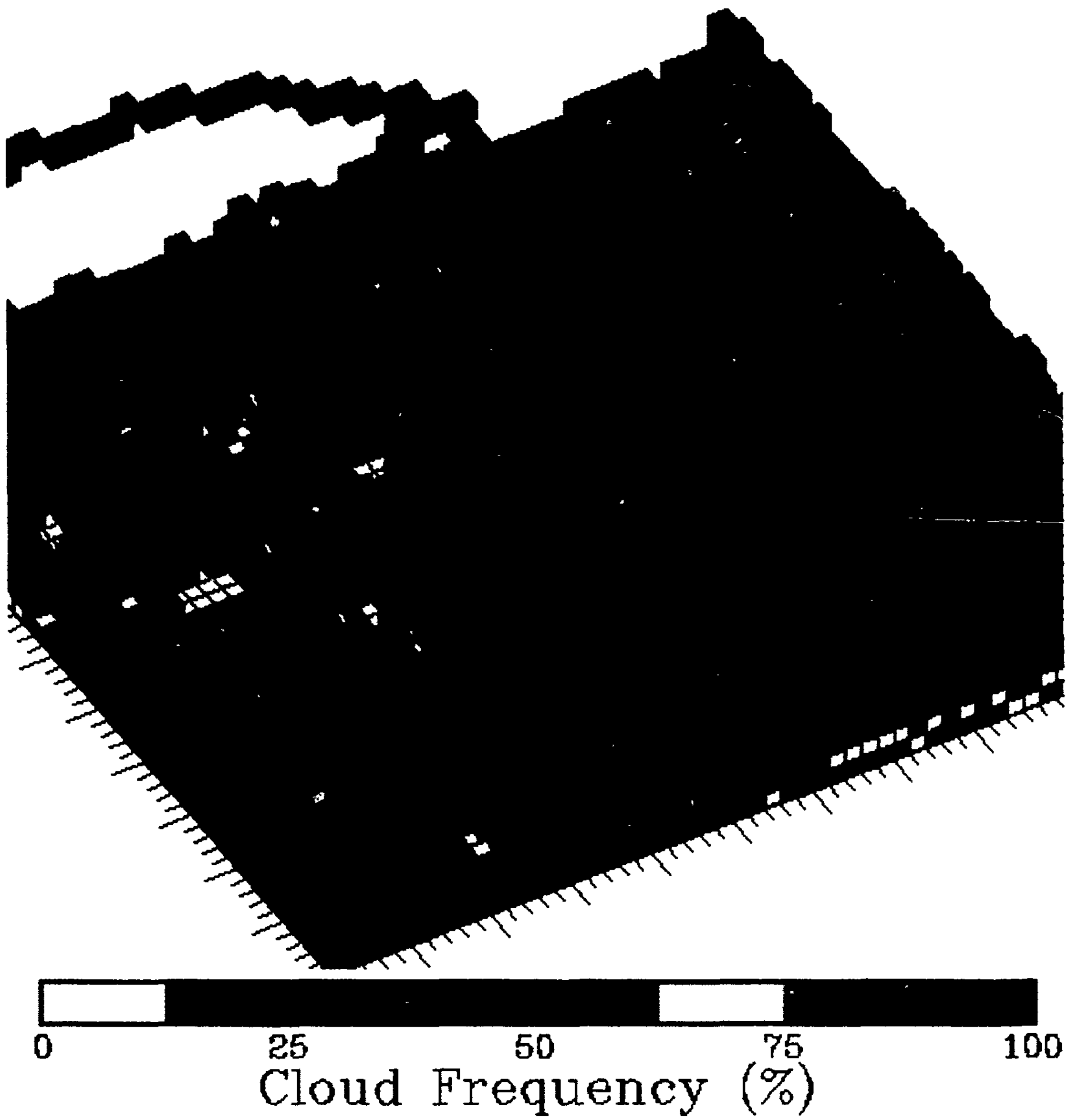

Figure 4 31: 15Z December 1990 3-D IR cloud climatology of Nicaraguan region at 25% cloud threshold.

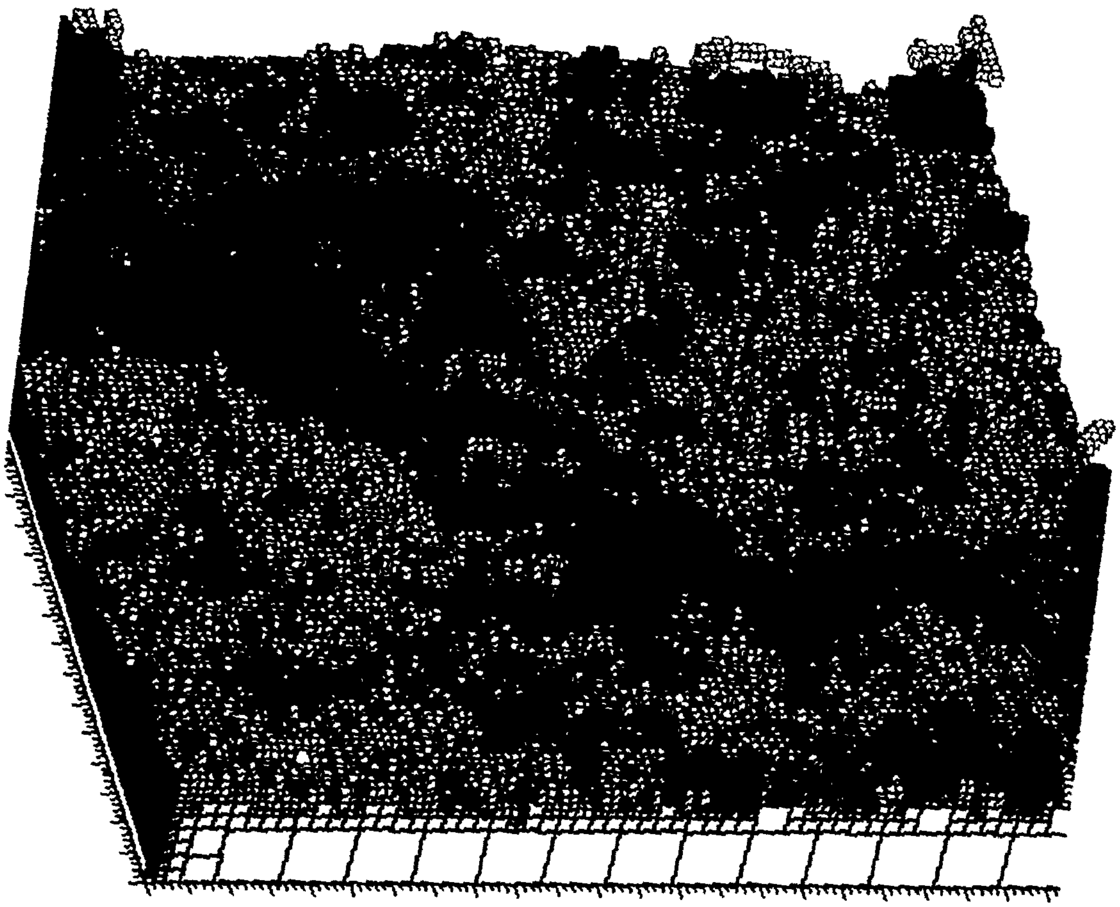

Figure 4 32· December 1990 15Z to 18Z three-dimensional topographic change in cloud frequency-of-occurrence. Climatologies were IR with threshold 9 counts above background. White: 15Z frequency Orange: 18Z frequency Yellow: intersection

4.3 VISUALIZATION OF CUMULUS CLOUD GROWTH

The octree structure is well-suited to many different types of visualization. One data set that works especially well with octrees is the representation of cumulus cloud growth. Since cumulus cloud systems are generally coherent in nature, their relatively smooth surfaces are ideal for efficient octree encoding. Regional atmospheric modelers and forecasters alike could benefit from a 3-D octree visualization of cloud growth between two valid times.

4.3.1 DISPLAYING GROWTH WITH DIFFERENT COLORS

Cloud growth can be displayed by using the same technique demonstrated in the previous section: merging two images and using color to differentiate between them. The original GOES images used for this example were extracted from the same sector of a larger image at 15Z and 18Z of the same day (see Figures 4.33 and 4.34). For this application, GOESVOX was modified to accept 64 x 64 pixel images, and output a 64 x 64 x 64 dimension array. The resulting octree display is shown in Figure 4.35. The color scheme is again: 15Z cloud in white, 18Z cloud in orange, and their intersection in yellow.

More information on the growth of the cumulus system can be obtained through slicing the image along several planes. Figures 4.36-4.39 show how clipping can reveal the inner structure of the system's growth. Obviously, this system was slightly taller at 15Z, since the octree is capped by white cubes, but this white layer is shallow, indicating the cloud system has grown significantly around the periphery without losing very much in height.

This octree visualization is again quite simple to interpret and comprehend. It easily shows the spatial change in both vertical and horizontal dimensions, with very little time required for image interpretation. By calibrating the brightness temperatures to an atmospheric sounding, specific height values could be attached to each cube, making the analysis even more valuable.

4.3.2 DISPLAYING GROWTH WITH A TRANSPARENT SHELL

Another method of displaying the growth of the cloud system is through the use of a transparent shell. This display is not an inherent feature of the VOXEL software, but can be created by letting VOXEL and IMX each handle part of the task. Similar to "wire frame" boundaries already discussed, the transparent shell is created by using VOXEL to display and save images of the cloud both with and without the 18Z growth (using the same magnification and rotation). Then IMX can take the image with growth, and make a graphic overlay from just the black edges of the cubes.

Figure 4.40 shows the result of displaying just the 15Z portion of the cloud in white and yellow, and then overlaying a red graphic of the 18Z growth to create the appearance of a transparent "shell of growth". Actually, a black graphic of the 15Z cube edges had to be re-overlaid to prevent *all* the edges from appearing red.

In this particular example, the red "shell" may not be as good a representation as a solid orange surface, since the immense number of cubes drawn make it difficult to discern the shell's three-dimensional placement. A display with fewer cubes would lead to less clutter, and the transparent shell could become extremely useful.

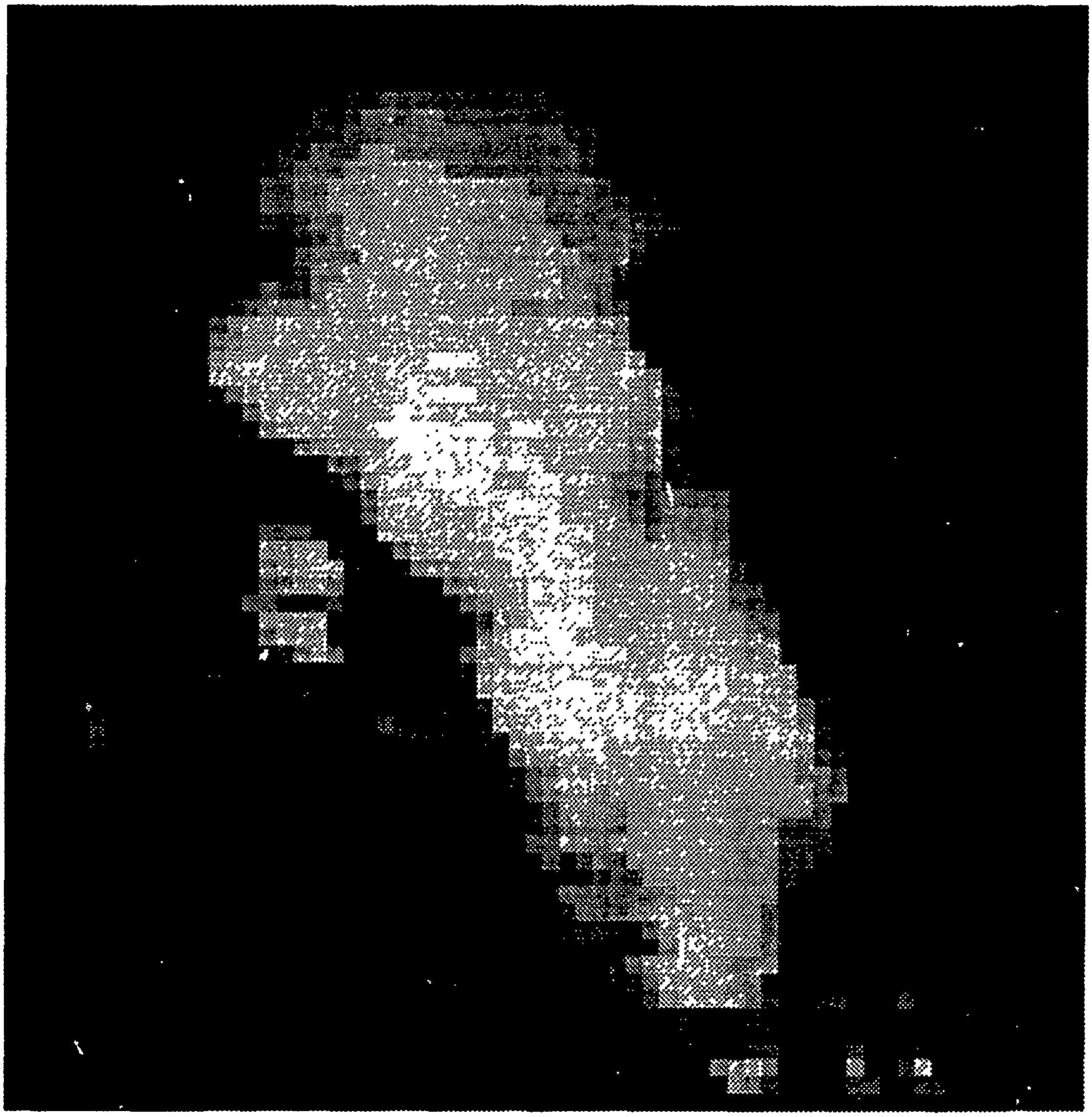

Figure 4.33: IR Close-up of cumulonimbus system at 15Z.

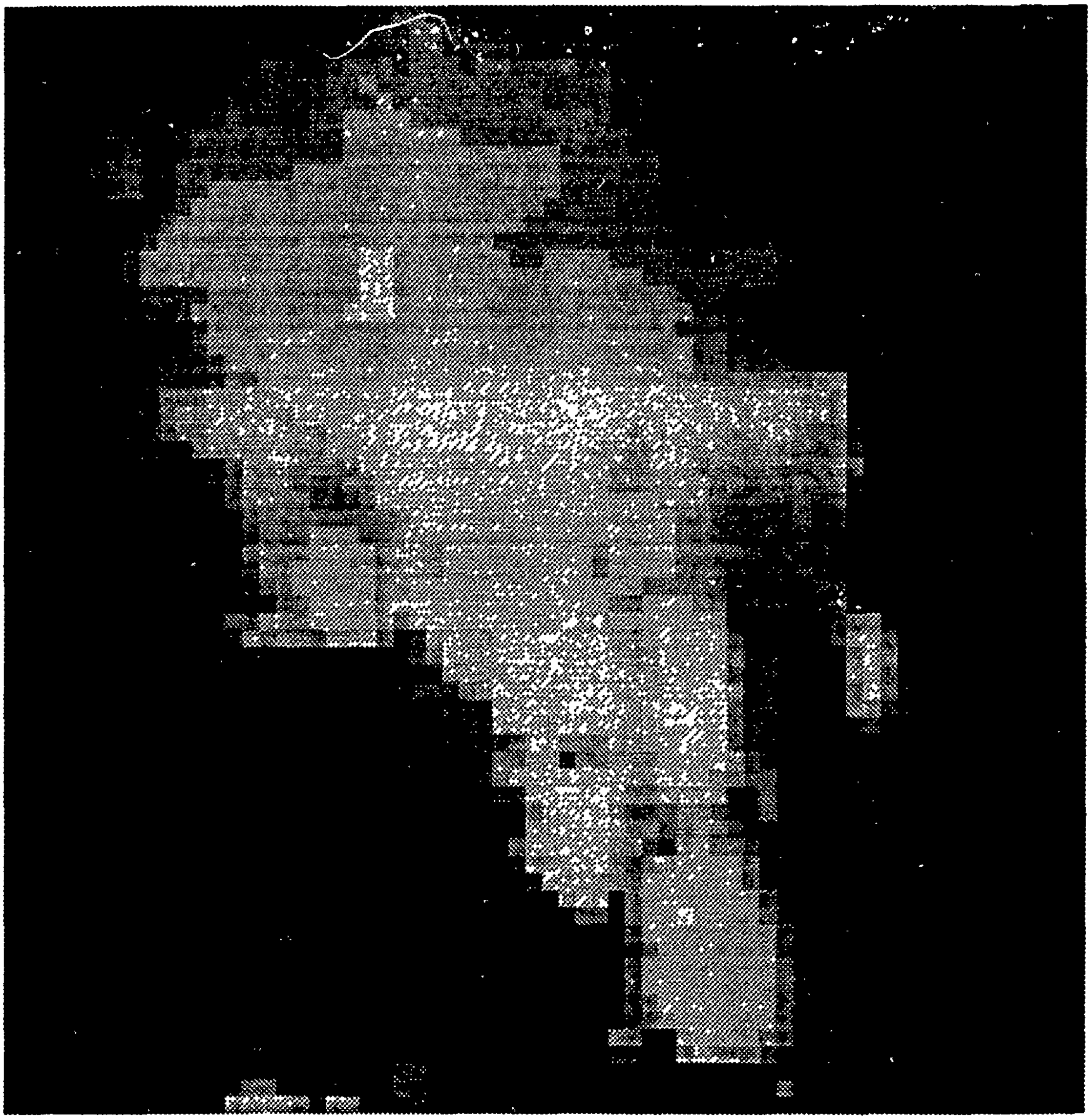

Figure 4.34: IR Close-up of cumulonimbus system at 18Z (same sector as Fig. 4.33).

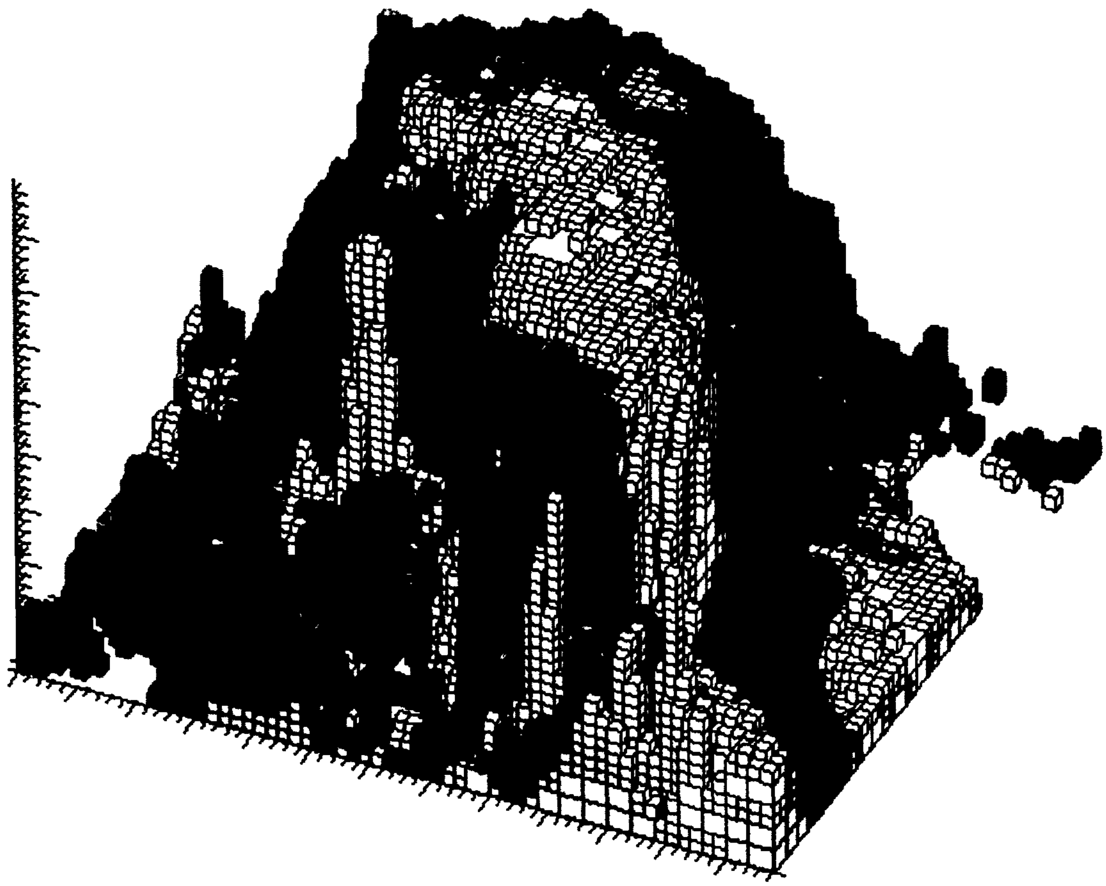

Figure 4.35: 3-D visualization of merged 15Z and 18Z cumulonimbus systems.
White: 15Z cloud Orange: 18Z cloud Yellow: intersection of clouds

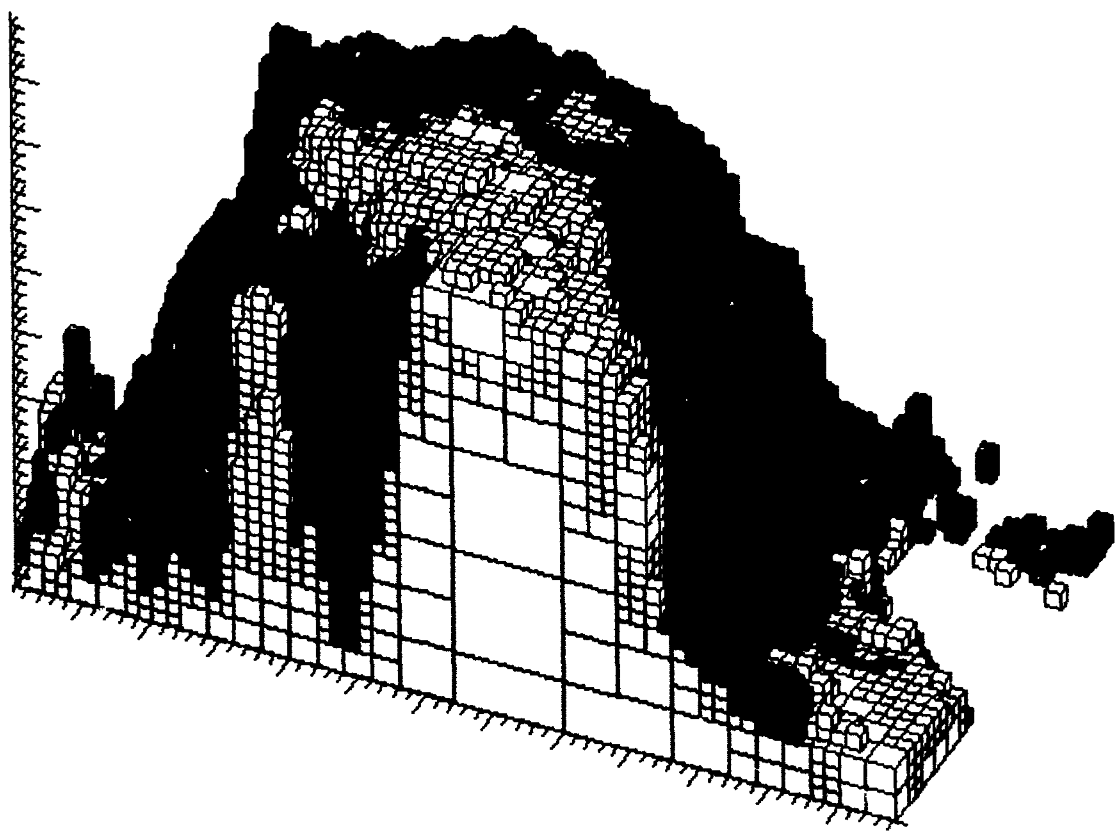

Figure 4.36: 3-D merged 15Z and 18Z cumulonimbus, clipped 26 rows from the south edge.
White· 15Z cloud Orange: 18Z cloud Yellow: intersection

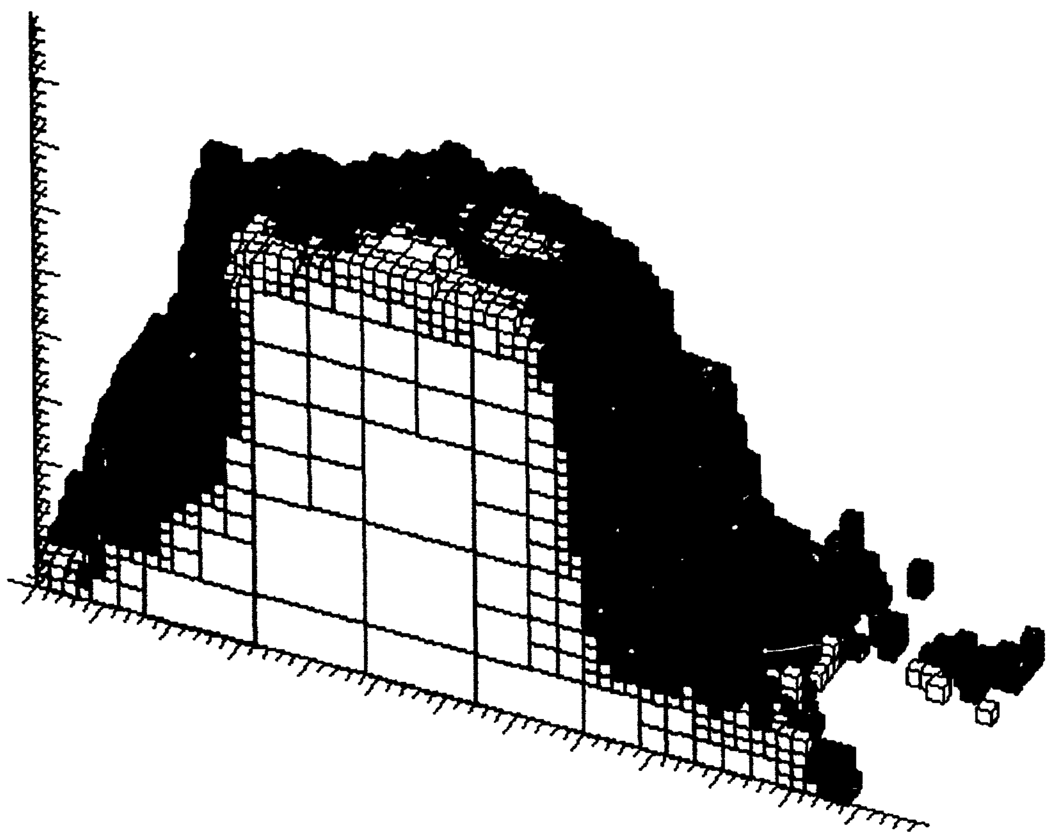

Figure 4.37: 3-D merged 15Z and 18Z cumulonimbus, clipped 40 rows from the south edge.
White: 15Z cloud Orange: 18Z cloud Yellow: intersection

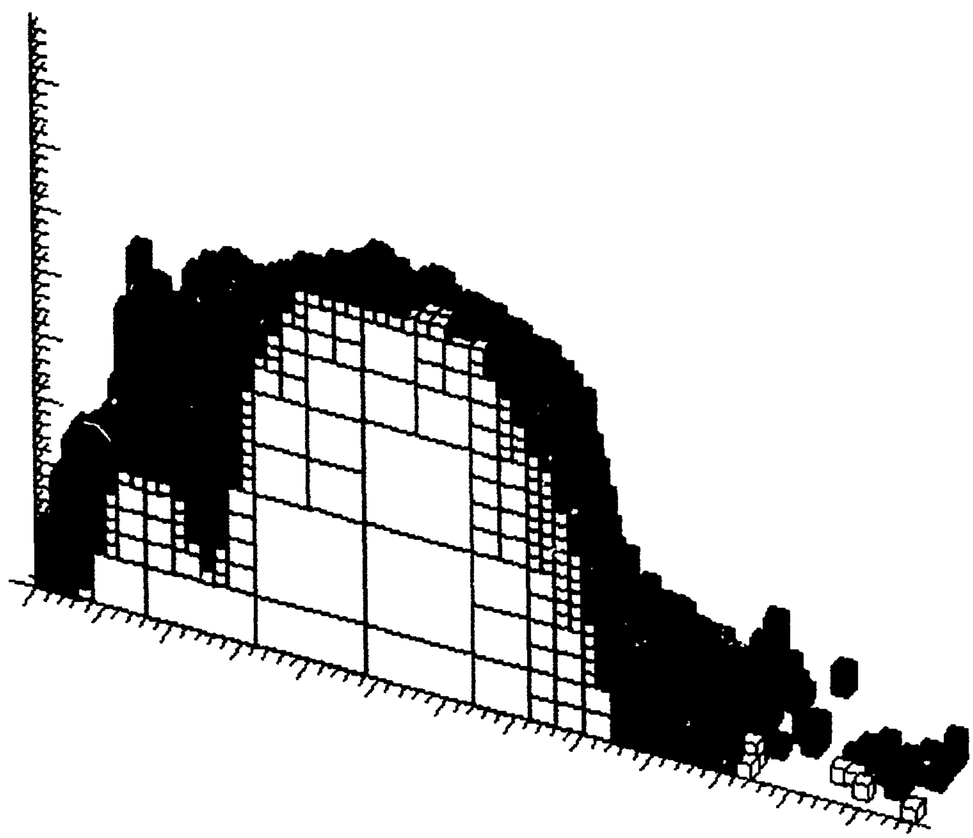

Figure 4.38: 3-D merged 15Z and 18Z cumulonimbus, clipped 52 rows from the south edge.
White: 15Z cloud Orange: 18Z cloud Yellow: intersection

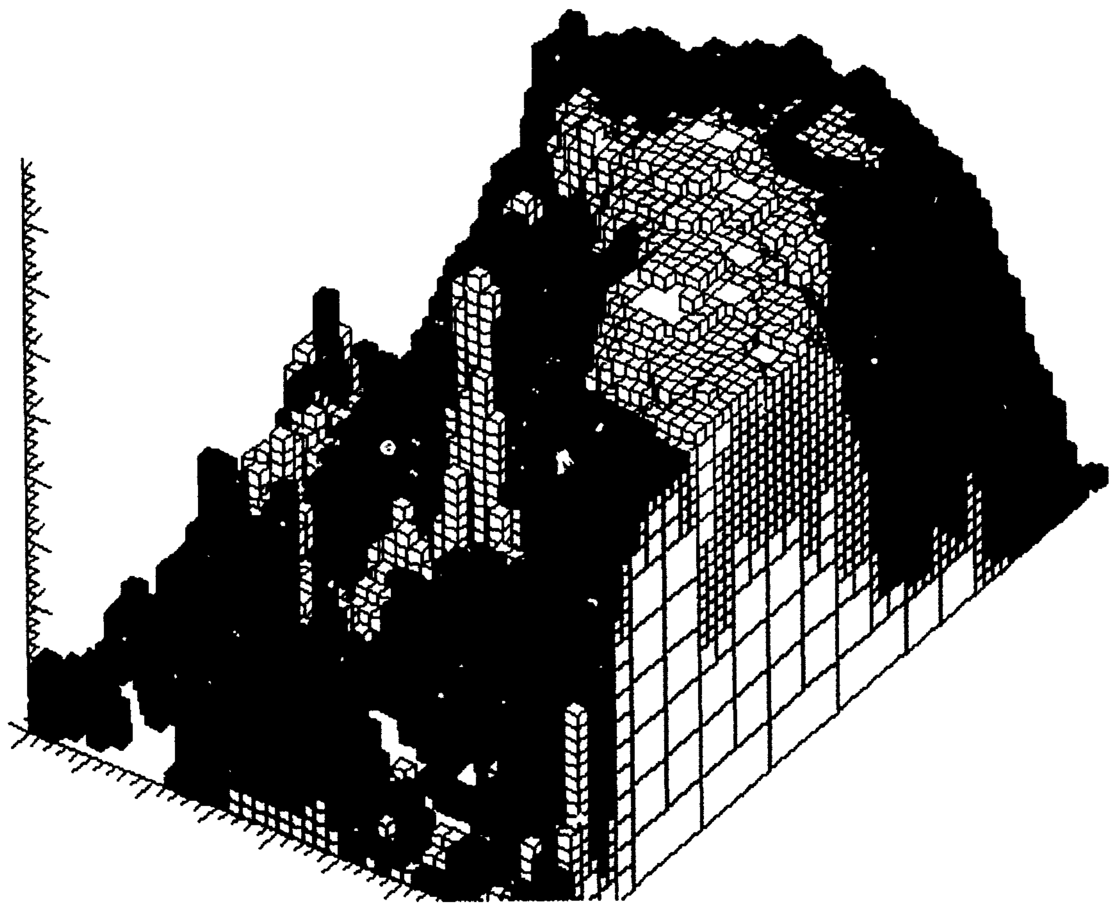

Figure 4.39: 3-D merged 15Z and 18Z cumulonimbus, clipped 42 rows from the west edge.
White: 15Z cloud Orange: 18Z cloud Yellow: intersection

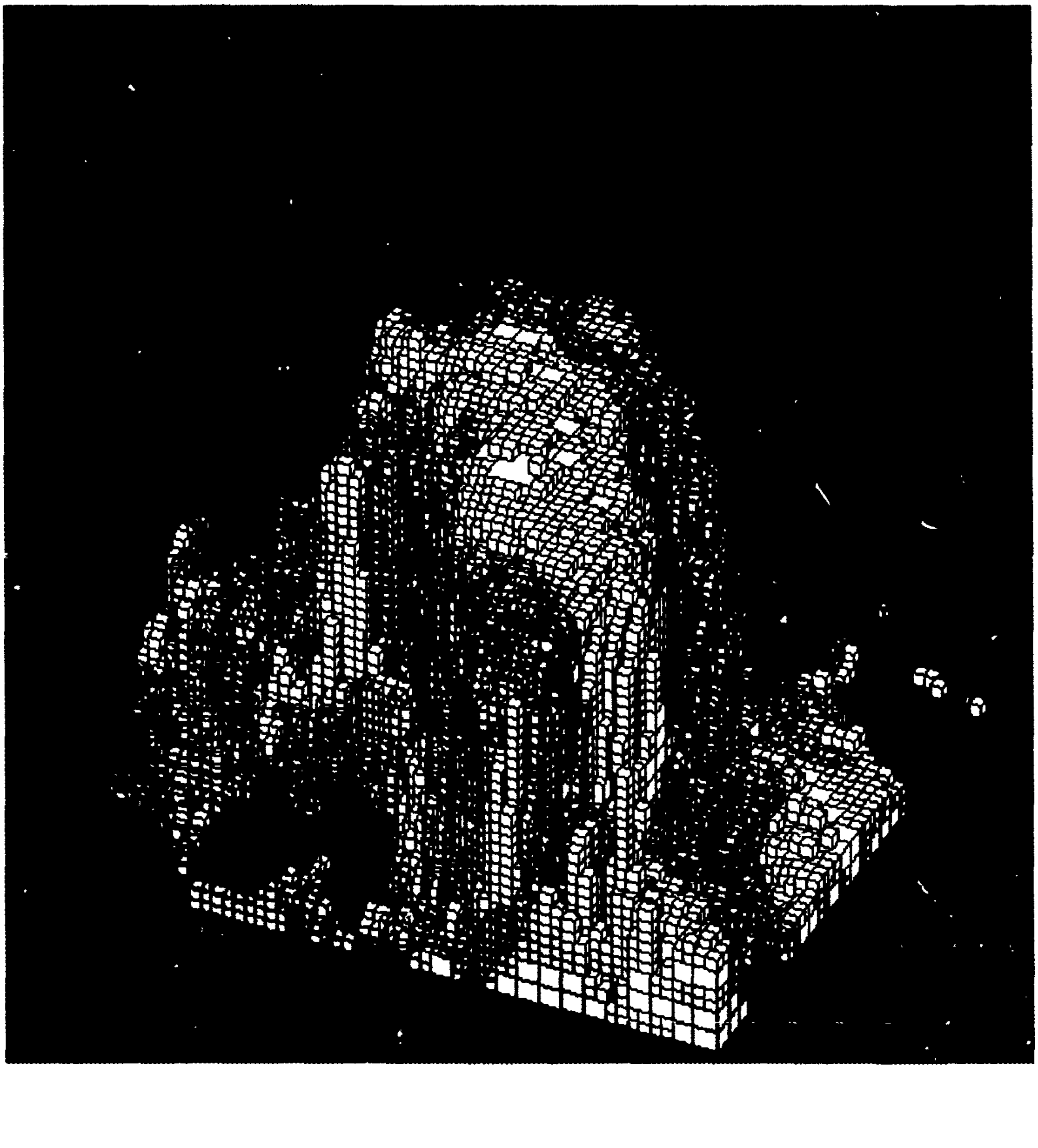

Figure 4.40: 3-D 15Z cumulonimbus with red 18Z "wire frame" overlay.
White: 15Z cloud Yellow: intersection

Chapter 5

CONCLUSIONS AND FUTURE WORK

The usefulness of the octree structure[10] as a tool to visualize GOES cloud data has been investigated in this thesis. Some of the advantages to octree visualization include the ability to retain the inner structure of a volume of data, the data storage efficiency (depending on the data set), and the simplicity of the cubic building block which makes program code short and straightforward.

When teamed with the VOXEL software, octrees become a powerful tool for visualizing any three-dimensional data set. The octrees can be both magnified and rotated to attain an optimum viewpoint. And to examine the inner structure, both slicing and thresholding can be utilized. The looping feature allows differing views to be easily compared to one another, and the image save command allows exporting views to IMX for further image processing.

The GOES cloud data examples showed that merely converting a 2-D data set to topographic format would be beneficial, especially to the layman. A three-dimensional display may be easier to relate to, allowing better comprehension, and possibly a more accurate analysis. Generally, octree-encoding this type of data is very efficient. Even the climatology data, which is far from smooth, could be reduced from a Cartesian array of 513 memory blocks to an octree of 165 blocks.

The 3-D climatologies demonstrated the real power of VOXEL analysis by using clipping and thresholding to examine areas of interest. By attaching temperature and height values to each cloud-frequency threshold, this type of display could be quite valuable to airlines (for planning f[illegible] routes) and forecasters (who need to know patterns of

[10] As implemented in the VO[illegible] [illegible]ware package.

convective development). Though very effective visualizations could be created from this data, it was not well suited to efficient encoding as very few of the cubes found large enough chunks of homogeneous data to grow beyond a dimension of 1. Consequently, the memory savings were only between 15 and 20%.

Octrees are also useful in displaying differences in data sets. Setting different valid times to different colors allows meaningful displays of both changes in climatological data sets and growth in a cumulus cloud system. Both types of applications would be useful to weather forecasters in an operational setting. Three-dimensional clouds are especially efficient in the octree encoding process, as their surfaces are generally coherent in nature. This smoothness requires fewer small cubes for showing detail, and allowed the octree encoded example[11] to occupy less than 15% of the memory required by the Cartesian array.

Clearly, the octree can be a very powerful and useful tool for analysis of cloud data from satellite. Another application that should be explored is that of displaying output from regional atmospheric models. The octree could be used to represent any number of output parameters, such as vertical velocity, temperature, humidity, pressure surfaces, or even combinations of these. Since these atmospheric models are generally low resolution, they would be a perfect fit to octree representation.

The VOXEL program itself should be modified to become even more powerful and versatile. To quantify the rotation, the display should include an output of degrees from each major axis. A method of overlaying line graphics of geopolitical boundaries would make displays more meaningful and readable. An even more efficient version of octree could be implemented, called *linear octree*[12]. The VOXEL display window should be increased in size (from its prese· ? x 512) to allow better graphic resolution. An upgrade just implemented (T. Brubake ial communication) has increased the maximum VOXEL octree dimensions to z.. As faster computer processors are installed, the maximum dimensions should be further increased to allow even larger data sets to be used.

---

[11] In Section 4.3

[12] Explained in Li, 1992

REFERENCES

Ahuja, N., and J. Veenstra, 1989: Generating Octrees from Object Silhouettes in Orthographic Views, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, No. 2, 137-149.

Anand, S., and K. Scott, 1991: An Algorithm for Converting the Boundary Representation of a CAD model to its Octree Representation, *Computers & Industrial Engineering*, **21**, No. 1, 343-351.

Clark, J.D., 1983: *The GOES Users Guide.* Washington, D.C. : U.S. Dept. of Commerce, NOAA, NESDIS, p. 7-3

Conti, P., N. Hitschfeld, and W. Fitchner, 1991: Ω-- An Octree-Based Mixed Element Grid Allocator for the Simulation of Complex 3-D Device Structures, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **10**, No. 10, 1231-1241.

Faverjon, B., 1984: Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator, *IEEE International Conference on Robotics*, pp. 504-512.

Gibson, H., 1988: Cloud and Convection Frequencies Relative to Small-Scale Geographic Features, Master's Thesis, Atmospheric Science Department, Colorado State University

Hull, M., J. Frazer, and R. Millar, 1990: Octree-Based Modeling of Computed-Tomography Images, *IEE Proceedings. I, Communications Speech and Vision*, **137**, No. 3, 118-133.

Hunter, G., 1978: Efficient Computation and Data Structures for Graphics, Ph.D. dissertation, Electrical Engineering and Computer Science Department, Princeton University.

Huonder, R., 1990. Visualization of Three-Dimensional Scientific Data Using Octrees, Master's Thesis, Computer Science Department, Colorado State University.

Kidder, S., and T. H. Vonder Haar, 1992: *Satellite Meteorology: An Introduction.* Academic Press, Inc., pp. 4-34,4-35,4-70

Li, R., 1992: Building Octree Representations of Three-Dimensional Objects in CAD/CAM by Digital Image Matching Techniques, *Photogrammetric Engineering and Remote Sensing*, **58**, No. 12, 1685-1691

Meagher, D., 1984: Interactive Solids Processing for Medical Analysis and Planning, *Proc. Comput. Graphics*, NCGA.

Rudloff, W., 1981: *World-Climates with Tables of Climatic Data and Practical Suggestions.* Wissenschlaftliche Verlagsgesellschaft mbH Stuttgart, p. 21

Schwerdtfeger, W., 1976: *Climates of Central and South America.* Elsevier Scientific Publishing Company, pp. 414-417

Wilhelms, J. and A. Van Gelder, 1992: Octrees for Faster Isosurface Generation, *ACM Transactions on Graphics,* **11**, No. 3, 201-227

APPENDIX

GOESVOX CONVERSION PROGRAM

The following pages show three versions of the GOESVOX software which was used to convert the GOES imagery to 3-D data arrays that VOXEL could read. The first is the original program (not yet entirely functional) written by Jan Behunik at CIRA. The second version labeled GOESVOX128 includes modifications to read in radiosonde data to enable an IR image to be converted from brightness counts directly to height coordinates. The third version is the one used to create 16-level cloud climatologies from sixteen 128 x 128 pixel images (at sixteen different thresholds). Note that the VOXEL program expects the X axis to point east, the Y axis to point up, and the Z axis to point south (the conventional directions in most computer graphics work).

The subroutine VOXEL_WRITE_FOR at the end of the programs is actually a subroutine in the VOXEL library that takes the byte data in the three-dimensional array BOCT(X,Y,Z) and writes it into a specially formatted data file that VOXEL can read.

```
      PROGRAM GOESVOX
C*************************************************************************
C
C      PROGRAM GOESVOX
C
C      This program reads (a portion of) a GOES image and writes it as
C      a 3-D Voxel array.
C
C*************************************************************************
      LOGICAL LOUT/.FALSE./
      CHARACTER CIGOES*9
      INCLUDE 'IRIS$INC:HIGHIMG.FIN/LIST'
C
C...      Get image name and data base entry.
C
      TYPE *, ' Enter input GOES image name (no .ext):'
      ACCEPT 501, CIGOES
501   FORMAT(A)
      CALL OPENDB('PARAM', ISTATRM)
      IF (ISTATRM .NE. 0) CALL FRIMERR(' Error opening database.',
     +      ISTATRM)
      CALL OLDIMAGE(1, CIGOES, STATUS)
      IF (.NOT. SUCCESS(STATUS)) THEN
       CALL TYPE_ERROR(MSG3, STATUS)
       IF (FATAL(STATUS)) CALL EXIT
      ENDIF
C
C...      Determine center point of region to extract, process.
C
      TYPE *, ' Enter center point of region to extract from GOES'
      TYPE *, ' image (lat, -lon):'
      ACCEPT *, ALAT, ALON
C
      CALL INITTRAN(IPTR1)
      CALL SETCOORD(IPTR1, 'EARTH', 'GOES_AUTO', IMG_DATE(1),
     +      IMG_TIME(1), IMG_SOURCE(1), IMG_DATE(1), IMG_TIME(1),
     +      IMG_SOURCE(1), ISTAT)
      IF (ISTAT .GT. 1) CALL FERROR2(' Error calling SETCOORD.',
     +      ISTAT)
      CALL TRANS(IPTR1, ALON, ALAT, XIMG, YIMG)
      CALL CLRTRAN(IPTR1)
C
C...      Get Voxel output specs.
C
      TYPE *, ' Enter X, Y, Z dimensions of output voxel display:'
      ACCEPT *, IVX, IVY, IVZ
      TYPE *, ' Enter horizontal, vertical resolution of voxel'
      TYPE *, ' display (km):'
      ACCEPT *, HRES, VRES
C
C...      Compute boundaries of GOES data to process.
C
      XFACT = IMG_IMGSCT(1,1) / HRES
      YFACT = ABS(IMG_IMGSCT(2,1) / HRES
```

```
      XLEDGE = XIMG - (IVX / 2.) * HRES
      XREDGE = XIMG + (IVX / 2.) * HRES
      YUEDGE = YIMG - (IVY / 2.) * HRES
      YLEDGE = YIMG + (IVY / 2.) * HRES
      IF (XLEDGE .LT. IMG_IMGSCT(3,1)) LOUT = .TRUE.
      IF (XREDGE .GT. IMG_IMGSCT(3,1) + IMG_IMGSCT(4,1) * 2)
     +         LOUT = .TRUE.
      IF (YUEDGE .LT. IMG_IMGSCT(5,1)) LOUT = .TRUE.
      IF (YLEDGE .GT. IMG_IMGSCT(5,1) - IMG_IMGSCT(6,1) * 2)
     +         LOUT = .TRUE.
      IF (LOUT) THEN
       TYPE *, ' Voxel area extends outside image. Terminating.'
       CALL EXIT
      ENDIF
      IRS = NINT((YUEDGE - IMG_IMGSCT(5,1)) / ABS(IMG_IMGSCT(2,1)) + 1
      NREC = NINT((YLEDGE - YUEDGE) / ABS(IMG_IMGSCT(2,1))
      IES = NINT((XLEDGE - IMG_IMGSCT(3,1)) / IMG_IMGSCT(1,1)) + 1
      NEL = NINT((XREDGE - XLEDGE) / IMG_IMGSCT(1,1))
C
      TYPE *, ' Enter cloud threshold brightness count.'
      ACCEPT *, ITHR
C
C...        Read rawinsonde data.
C
      CALL READRAW
C
      CIEXT = CIGOES//'.IMG'
      OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
C
C...        Read and process each GOES scan line.
C
      ICY = 1
      DO 10 IREC = 1, NREC
       READ (1 ' IREC+IRS-1) IBBUF
       ICX = 1
       DO 20 IEL = IES, IES+NEL-1
        INT = IBBUF(IEL)
        IBRC = IAND(INT, 255)
        IF (IBRC .GE. ITHR) THEN
C
C...        IR count to TBB.
C
         CALL TEMP
C
C...        TBB to cloud top height.
C
         DO 30 L = 2, NRLEV
          IF (RT(L) .LE. TBB) THEN
             CTH(IN) = RHT(L) - ((RT(L) - TBB) * (RHT(L) - RHT(L-1))
                  / (RT(L) - RT(L-1)))
             GO TO 35
          ENDIF
30        CONTINUE
        ELSE
```

```
C***          Null value in octree for no-cloud!
        ENDIF
        IZ = NINT(CTH / VRES)
        IF (YFACT .GE. 1.) THEN
         DO ITY = 1, YFACT
          IY = ITY + ICY - 1
          IF (XFACT .GE. 1.) THEN
           DO ITX = 1, XFACT
             IX = ITX + ICX - 1
             DO ITZ = 5, IZ
              BOCT(IX,IY,IZ) = -126
             ENDDO
           ENDDO
          ENDIF
         ENDDO
        ENDIF
        ICX = ICX + XFACT
20       CONTINUE
       ICY = ICY + YFACT
10      CONTINUE
```

```
      PROGRAM GOESVOX128
C*****************************************************************************
C
C     PROGRAM GOESVOX128
C
C     This program reads a 128 X 128 GOES image and writes it as
C     a 3-D Voxel array.
C
C     Variables used:
C     CIGOES : Name of Satellite image (no .ext)
C     CTH : Cloud Top Height
C     DEWPT : Array of DEWPoinT temperatures from sounding
C     HRES : Horizontal Resolution of Voxel
C     IBRC : BRightness Count
C     IEL : reference ELement
C     IES : Element Start address
C     IMG_IMGSCT : Image Sector array
C     IRS : Record Start address
C     ITHR : Cloud THReshold brightness count
C     IVX : Voxel X dimension
C     IVY : Voxel Y dimension
C     IVZ : Voxel Z dimension
C     NREC : # of RECords (Y-direction)
C     NEL : # of ELements (X-direction)
C     RTEMP : Reference Temperature in degrees Celcius (array from sounding)
C     LEVEL : Reference HeighT in meters from sounding array
C     TBB : BlackBody Temperature in degrees Celcius
C     VRES : Vertical Resolution of Voxel
C     XFACT : # of Voxel Blocks in X-direction
C     XLEDGE : Left EDGE of Voxel display
C     XREDGE : Right EDGE of Voxel display
C     YFACT : # of Voxel Blocks in Y-direction
C     YTEDGE : Top EDGE of Voxel display
C     YBEDGE : Bottom EDGE of Voxel display
C
C*****************************************************************************
      LOGICAL SUCCESS,STATUS,FATAL
      CHARACTER CIGOES*9,STNID*3,CIEXT*13
      REAL RTEMP(100),LEVEL(100),DEWPT(100)
      BYTE BOCT(128,128,16),IBBUF(128),IVX,IVY,IVZ
      INCLUDE 'IRIS$INC:HIGHIMG.FIN/LIST'
C
C           Get image name and data base entry
C
      TYPE *, ' Enter input GOES image name (no .ext):'
      ACCEPT 501, CIGOES
501   FORMAT(A)
      CALL OPENDB('PARAM', ISTATRM)
      IF (ISTATRM .NE. 0) CALL FRIMERR(' Error opening database ',
     +       ISTATRM)
      CALL OLDIMAGE(1, CIGOES, STATUS)
      IF (.NOT. SUCCESS(STATUS)) THEN
       CALL TYPE_ERROR(MSG3, STATUS)
       IF (FATAL(STATUS)) CALL EXIT
```

```
      ENDIF
C
C...        Determine center point of region to extract, process.
C
      CALL INITTRAN(IPTR1)
      CALL SETCOORD(IPTR1, 'EARTH', 'GOES_AUTO', IMG_DATE(1),
     +      IMG_TIME(1), IMG_SOURCE(1), IMG_DATE(1), IMG_TIME(1),
     +      IMG_SOURCE(1), ISTAT)
      IF (ISTAT .GT. 1) CALL FERROR2(' Error calling SETCOORD.',
     +      ISTAT)
C
C...        Get Voxel output specs.
C
      IVX = 128
      IVY = 128
      IVZ = 16
      DATA VRES /1/
C
C...        Compute boundaries of GOES data to process.
C
      TYPE *, ' Enter cloud threshold brightness count:'
      ACCEPT *, ITHR
C
C...        Read rawinsonde data.
C
      TYPE *, ' Enter 3-letter Station ID:'
      ACCEPT 502, STNID
502   FORMAT(A)
      CALL READ_RAOB(NUMLEV,LEVEL,RTEMP,DEWPT,STNID)
      CIEXT = CIGOES//'.IMG'
      OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
C
C...        Read and process each GOES scan line.
C
      ICY = 1
      DO I = 1, 128
       TYPE *, 'ICY =',ICY
       READ (1 ' I) IBBUF
       ICX = 1
       DO J = 1, 128
C        TYPE *, 'ICX =',ICX
        INT = IBBUF(J)
        IBRC = IAND(INT, 255)
C        TYPE *, 'IBRC =',IBRC
        IF (IBRC .GE. ITHR) THEN
C
C...        IR count to TBB.
C
         CALL IRTEMP(IBRC,TBB)
C
C...        TBB to cloud top height.
C
         DO L = 2, NUMLEV
          IF (RTEMP(L) .LE. TBB) THEN
```

```
          CTH=LEVEL(L)-((RTEMP(L)-TBB)*(LEVEL(L)-
     +         LEVEL(L-1))/(RTEMP(L)-RTEMP(L-1)))
          GO TO 35
        ENDIF
       ENDDO
C
C***        TBB colder than any level in sounding!
C
       TYPE *,'TBB found that is colder than any level in sounding'
       CALL EXIT
35      CONTINUE
      ELSE
C
C***        Null value in octree for no-cloud!
C
       CTH=0.
      ENDIF
      ITY = 129 - ICY
      IZ = NINT(CTH / (VRES*1000.))
      IF(IZ.GT.16) IZ=16
      DO ITZ = 1, IZ
       BOCT(ICX,ITY,ITZ) = 64
      ENDDO
      ICX = ICX + 1
     ENDDO
     ICY = ICY + 1
    ENDDO
    CALL VOXEL_WRITE_FOR('SATD.DAT',BOCT,IVX,IVY,IVZ)
    STOP
    END
```

```
      PROGRAM GOESVOX128
C*****************************************************************************
C
C     PROGRAM GOESVOX128
C
C     This program reads 16 128 X 128 GOES images and writes them as
C     a 16-level 3-D Voxel array.
C
C     Variables used:
C     CIGOES : Name of Satellite image (no .ext)
C     CTH : Cloud Top Height
C     DEWPT : Array of DEWPoinT temperatures from sounding
C     HRES : Horizontal Resolution of Voxel
C     IBRC : BRightness Count
C     IEL : reference ELement
C     IES : Element Start address
C     IMG_IMGSCT : Image Sector array
C     IRS : Record Start address
C     ITHR : Cloud THReshold brightness count
C     IVX : Voxel X dimension
C     IVY : Voxel Y dimension
C     IVZ : Voxel Z dimension
C     NREC : # of RECords (Y-direction)
C     NEL : # of ELements (X-direction)
C     RTEMP : Reference Temperature in degrees Celcius (array from sounding)
C     LEVEL : Reference HeighT in meters from sounding array
C     TBB : BlackBody Temperature in degrees Celcius
C     VRES : Vertical Resolution of Voxel
C     XFACT : # of Voxel Blocks in X-direction
C     XLEDGE : Left EDGE of Voxel display
C     XREDGE : Right EDGE of Voxel display
C     YFACT : # of Voxel Blocks in Y-direction
C     YTEDGE : Top EDGE of Voxel display
C     YBEDGE : Bottom EDGE of Voxel display
C
C*****************************************************************************
      LOGICAL SUCCESS,STATUS,FATAL
      CHARACTER CIGOES*9,STNID*3,CIEXT*13
      REAL RTEMP(100),LEVEL(100),DEWPT(100)
      BYTE BOCT(128,16,128),IBBUF(128),IVX,IVY,IVZ
      INCLUDE 'IRIS$INC:HIGHIMG.FIN/LIST'
C
C...        Get image name and data base entry.
C
      TYPE *, ' Enter 1st input GOES image name (no .ext):'
      ACCEPT 501, CIGOES
501   FORMAT(A)
      IVX = 128
      IVY = 16
      IVZ = 128
      DATA VRES /1/
C      TYPE *, ' Enter cloud threshold brightness count:'
C      ACCEPT *, ITHR
C
```

```
C.          Read rawinsonde data.
C
C      TYPE *, ' Enter 3-letter Station ID:'
C      ACCEPT 502, STNID
C502   FORMAT(A)
C      CALL READ_RAOB(NUMLEV,LEVEL,RTEMP,DEWPT,STNID)
      CIEXT = CIGOES//'.IMG'
      OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
C
C.          Read and process each GOES scan line.
C
      ICY = 1
      ICZ = 1
      DO I = 1, 128
       TYPE *, 'ICZ =',ICZ
       READ (1 ' I) IBBUF
       ICX = 1
       DO J = 1, 128
C        TYPE *, 'ICX =',ICX
        INT = IBBUF(J)
        IBRC = IAND(INT, 255)
        CTH=IBRC
          BOCT(ICX,ICY,ICZ) = CTH
        ICX = ICX + 1
       ENDDO
       ICZ = ICZ + 1
      ENDDO
      CLOSE(UNIT = 1)
      TYPE *, ' Enter 2nd input GOES image name (no .ext):'
      ACCEPT 501, CIGOES
      CIEXT = CIGOES//'.IMG'
      OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
      ICY = 2
      ICZ = 1
      DO I = 1, 128
       TYPE *, 'ICZ =',ICZ
       READ (1 ' I) IBBUF
       ICX = 1
       DO J = 1, 128
        INT = IBBUF(J)
        IBRC = IAND(INT, 255)
        CTH=IBRC
          BOCT(ICX,ICY,ICZ) = CTH
        ICX = ICX + 1
       ENDDO
       ICZ = ICZ + 1
      ENDDO
      CLOSE(UNIT = 1)
      TYPE *, ' Enter 3rd input GOES image name (no .ext):'
      ACCEPT 501, CIGOES
      CIEXT = CIGOES//'.IMG'
      OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
      ICY = 3
      ICZ = 1
```

```
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 4th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 4
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 5th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 5
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
```

```
TYPE *, ' Enter 6th input GOES image name (no  ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 6
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 7th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 7
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 8th input GOES image name (no  ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 8
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
```

```
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 9th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 9
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 10th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 10
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 11th input GOES image name (no .ext) '
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 11
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
```

```
READ (1 ' I) IBBUF
ICX = 1
DO J = 1, 128
 INT = IBBUF(J)
 IBRC = IAND(INT, 255)
 CTH=IBRC
    BOCT(ICX,ICY,ICZ) = CTH
 ICX = ICX + 1
ENDDO
ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 12th input GOES image name (no ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 12
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 13th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 13
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 14th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
```

```
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 14
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 15th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1, FILE = CIEXT, STATUS = 'OLD', ACCESS = 'DIRECT')
ICY = 15
ICZ = 1
DO I = 1, 128
 TYPE *, 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
TYPE *, ' Enter 16th input GOES image name (no .ext):'
ACCEPT 501, CIGOES
CIEXT = CIGOES//'.IMG'
OPEN(UNIT = 1. FILE = CIEXT, STATUS = 'OLD'. ACCESS = 'DIRECT')
ICY = 16
ICZ = 1
DO I = 1, 128
 TYPE *. 'ICZ =',ICZ
 READ (1 ' I) IBBUF
 ICX = 1
 DO J = 1, 128
  INT = IBBUF(J)
  IBRC = IAND(INT, 255)
  CTH=IBRC
     BOCT(ICX,ICY,ICZ) = CTH
  ICX = ICX + 1
```

```
 ENDDO
 ICZ = ICZ + 1
ENDDO
CLOSE(UNIT = 1)
CALL VOXEL_WRITE_FOR('SATD.DAT',BOCT,IVX,IVY,IVZ)
STOP
END
```